



Introducción

Vibe Coding, SDD (Copilot Spec Kit), AGENTS.md y Agent Skills

Guía Integrada de Desarrollo de Software Asistido por IA

50 minutos

La Era del Desarrollo Asistido por IA

De los snippets de código a los agentes autónomos

5 minutos

La Evolución del Desarrollo de Software con IA

La Evolución del Desarrollo de Software con IA

- **Fase 1 (2018-2021):** Autocompletado inteligente - Sugerencias de líneas individuales basadas en contexto local

La Evolución del Desarrollo de Software con IA

- **Fase 1 (2018-2021):** Autocompletado inteligente - Sugerencias de líneas individuales basadas en contexto local
- **Fase 2 (2021-2024):** Generación de código - LLMs capaces de generar funciones completas a partir de descripciones

La Evolución del Desarrollo de Software con IA

- **Fase 1 (2018-2021):** Autocompletado inteligente - Sugerencias de líneas individuales basadas en contexto local
- **Fase 2 (2021-2024):** Generación de código - LLMs capaces de generar funciones completas a partir de descripciones
- **Fase 3 (2024-presente):** Agentes autónomos - Sistemas que planifican, ejecutan y verifican tareas complejas

La Evolución del Desarrollo de Software con IA

- **Fase 1 (2018-2021):** Autocompletado inteligente - Sugerencias de líneas individuales basadas en contexto local
- **Fase 2 (2021-2024):** Generación de código - LLMs capaces de generar funciones completas a partir de descripciones
- **Fase 3 (2024-presente):** Agentes autónomos - Sistemas que planifican, ejecutan y verifican tareas complejas

Cada fase ha ampliado el alcance de lo que la IA puede hacer, pero también ha introducido nuevos desafíos en control y predictibilidad.

El Flujo de Trabajo con IA Generativa



El Flujo de Trabajo con IA Generativa



El "muro de la complejidad" aparece cuándo:

- El contexto del proyecto excede la ventana del modelo
- Las dependencias entre componentes requieren coherencia global
- Los cambios en una parte afectan múltiples áreas del código

Desafíos: Consistencia, Predictibilidad y Mantenimiento

Problema	Descripción	Consecuencia
Consistencia	El modelo no recuerda decisiones anteriores entre sesiones	Código con estilos y patrones inconsistentes
Predictibilidad	Mismos prompts pueden generar resultados diferentes	Dificultad para reproducir y depurar
Mantenimiento	Código generado sin estructura clara	Deuda técnica acumulada rápidamente

Desafíos: Consistencia, Predictibilidad y Mantenimiento

Problema	Descripción	Consecuencia
Consistencia	El modelo no recuerda decisiones anteriores entre sesiones	Código con estilos y patrones inconsistentes
Predictibilidad	Mismos prompts pueden generar resultados diferentes	Dificultad para reproducir y depurar
Mantenimiento	Código generado sin estructura clara	Deuda técnica acumulada rápidamente

Estos problemas han motivado el desarrollo de metodologías estructuradas como Spec-Driven Development, archivos de contexto como AGENTS.md, y sistemas modulares como Agent Skills.

El Ecosistema del Desarrollo Asistido por IA

Cuatro dimensiones de interacción con agentes de IA

5 minutos

Los Cuatro Enfoques en Perspectiva

Enfoque	Dimensión Principal	Característica Clave
Vibe Coding	Exploración	Fluidez e intuición
Spec Kit (SDD)	Estructuración	Especificaciones formales
AGENTS.md	Contexto	Documentación unificada
Agent Skills	Ejecución	Capacidades especializadas

Vibe Coding

El enfoque fluido e intuitivo para desarrollo con IA

10 minutos

Qué es Vibe Coding?

Qué es Vibe Coding?

- Técnica de desarrollo asistida por IA donde se expresa intención en lenguaje natural

Qué es Vibe Coding?

- Técnica de desarrollo asistida por IA donde se expresa intención en lenguaje natural
- Enfoque chatbot-based para crear software

Qué es Vibe Coding?

- Técnica de desarrollo asistida por IA donde se expresa intención en lenguaje natural
- Enfoque chatbot-based para crear software
- El término "vibe" captura la naturaleza intuitiva y fluida

Qué es Vibe Coding?

- Técnica de desarrollo asistida por IA donde se expresa intención en lenguaje natural
- Enfoque chatbot-based para crear software
- El término "vibe" capture la naturaleza intuitiva y fluida
- El desarrollador se enfoca en expresar intención, la IA maneja implementación técnica

Qué es Vibe Coding?

- Técnica de desarrollo asistida por IA donde se expresa intención en lenguaje natural
- Enfoque chatbot-based para crear software
- El término "vibe" capture la naturaleza intuitiva y fluida
- El desarrollador se enfoca en expresar intención, la IA maneja implementación técnica
- Elimina la barrera de años de experiencia para construir aplicaciones funcionales

Ejemplo Práctico: Prompt de Vibe Coding

"Quiero una pagina de login con email y password,
un boton de 'olvide mi contrasena',
y que valide que el email tenga formato valido.
Hazlo con un diseno moderno y limpio."

Ejemplo Práctico: Código Generado por IA

```
1 const LoginScreen = () => {
2   const [email, setEmail] = useState('');
3   const [password, setPassword] = useState('');
4   const [emailError, setEmailError] = useState('');
5
6   const validateEmail = (email) => {
7     const re = /^[^@\s]+@[^\s]+\.\[^@\s]+\$/;
8     return re.test(email);
9   };
10
11  const handleLogin = () => {
12    if (!validateEmail(email)) {
13      setEmailError('Por favor ingresa un email válido');
14      return;
15    }
16    // Logica de login...
17  };
18
19  return (

```

Casos de Uso Ideales para Vibe Coding

Casos de Uso Ideales para Vibe Coding

- **Prototipado rápido** de ideas y conceptos

Casos de Uso Ideales para Vibe Coding

- **Prototipado rápido** de ideas y conceptos
- **Aprendizaje** de nuevas tecnologías o frameworks

Casos de Uso Ideales para Vibe Coding

- **Prototipado rápido** de ideas y conceptos
- **Aprendizaje** de nuevas tecnologías o frameworks
- **Tareas pequeñas** bien definidas y autocontenidoas

Casos de Uso Ideales para Vibe Coding

- **Prototipado rápido** de ideas y conceptos
- **Aprendizaje** de nuevas tecnologías o frameworks
- **Tareas pequeñas** bien definidas y autocontenidoas
- **Exploración** de soluciones alternativas

Casos de Uso Ideales para Vibe Coding

- **Prototipado rápido** de ideas y conceptos
- **Aprendizaje** de nuevas tecnologías o frameworks
- **Tareas pequeñas** bien definidas y autocontenidoas
- **Exploración** de soluciones alternativas
- **Generación inicial** de código que será refinado posteriormente

Casos de Uso Ideales para Vibe Coding

- **Prototipado rápido** de ideas y conceptos
- **Aprendizaje** de nuevas tecnologías o frameworks
- **Tareas pequeñas** bien definidas y autocontenidoas
- **Exploración** de soluciones alternativas
- **Generación inicial** de código que será refinado posteriormente
- **Proyectos personales** o experimentos sin requisitos estrictos de arquitectura

Limitaciones de Vibe Coding

Limitaciones de Vibe Coding

- Puede producir código que se siente productivo pero silenciosamente rompe la arquitectura

Limitaciones de Vibe Coding

- Puede producir código que se siente productivo pero silenciosamente rompe la arquitectura
- Las descripciones en lenguaje natural pueden ser ambiguas

Limitaciones de Vibe Coding

- Puede producir código que se siente productivo pero silenciosamente rompe la arquitectura
- Las descripciones en lenguaje natural pueden ser ambiguas
- Dificultad para mantener coherencia en proyectos grandes

Limitaciones de Vibe Coding

- Puede producir código que se siente productivo pero silenciosamente rompe la arquitectura
- Las descripciones en lenguaje natural pueden ser ambiguas
- Dificultad para mantener coherencia en proyectos grandes
- Falta de trazabilidad entre intención y código generado

Limitaciones de Vibe Coding

- Puede producir código que se siente productivo pero silenciosamente rompe la arquitectura
- Las descripciones en lenguaje natural pueden ser ambiguas
- Dificultad para mantener coherencia en proyectos grandes
- Falta de trazabilidad entre intención y código generado
- No adecuado para sistemas críticos que requieren estándares estrictos

Limitaciones de Vibe Coding

- Puede producir código que se siente productivo pero silenciosamente rompe la arquitectura
- Las descripciones en lenguaje natural pueden ser ambiguas
- Dificultad para mantener coherencia en proyectos grandes
- Falta de trazabilidad entre intención y código generado
- No adecuado para sistemas críticos que requieren estándares estrictos
- Código generado puede necesitar refactorización significativa

Recursos Adicionales - Vibe Coding

Recursos Adicionales - Vibe Coding

- **Vibe Coding - Wikipedia:** https://en.wikipedia.org/wiki/Vibe_coding

Recursos Adicionales - Vibe Coding

- **Vibe Coding - Wikipedia:** https://en.wikipedia.org/wiki/Vibe_coding
- **What is Vibe Coding? - IBM Think:** <https://www.ibm.com/think/topics/vibe-coding>

Recursos Adicionales - Vibe Coding

- **Vibe Coding - Wikipedia:** https://en.wikipedia.org/wiki/Vibe_coding
- **What is Vibe Coding? - IBM Think:** <https://www.ibm.com/think/topics/vibe-coding>
- **Vibe Coding in 2025: A Guide** (Plausible Futures): <https://plausiblerefutures.substack.com/p/vibe-coding-in-2025-a-technical-guide>

Recursos Adicionales - Vibe Coding

- **Vibe Coding - Wikipedia:** https://en.wikipedia.org/wiki/Vibe_coding
- **What is Vibe Coding? - IBM Think:** <https://www.ibm.com/think/topics/vibe-coding>
- **Vibe Coding in 2025: A Guide** (Plausible Futures): <https://plausiblerefutures.substack.com/p/vibe-coding-in-2025-a-technical-guide>
- **What is Vibe Coding? - Tanium:** <https://www.tanium.com/blog/what-is-vibe-coding/>

Spec-Driven Development (SDD)

El enfoque estructurado y sistemático para desarrollo con IA

20 minutos

Spec-Driven Development (SDD): Fundamentos

Spec-Driven Development (SDD): Fundamentos

- Metodología que transforma la interacción con agentes de IA

Spec-Driven Development (SDD): Fundamentos

- Metodología que transforma la interacción con agentes de IA
- Trata los agentes como programadores en pareja literalistas pero altamente capaces

Spec-Driven Development (SDD): Fundamentos

- Metodología que transforma la interacción con agentes de IA
- Trata los agentes como programadores en pareja literalistas pero altamente capaces
- Requiere especificaciones exhaustivas de antemano

Spec-Driven Development (SDD): Fundamentos

- Metodología que transforma la interacción con agentes de IA
- Trata los agentes como programadores en pareja literalistas pero altamente capaces
- Requiere especificaciones exhaustivas de antemano
- Proporciona imagen completa: qué construir, por qué importa y que NO construir

Spec-Driven Development (SDD): Fundamentos

- Metodología que transforma la interacción con agentes de IA
- Trata los agentes como programadores en pareja literalistas pero altamente capaces
- Requiere especificaciones exhaustivas de antemano
- Proporciona imagen completa: qué construir, por qué importa y que NO construir
- Las especificaciones técnicas actúan como fuente autoritativa de verdad

Spec-Driven Development (SDD): Fundamentos

- Metodología que transforma la interacción con agentes de IA
- Trata los agentes como programadores en pareja literalistas pero altamente capaces
- Requiere especificaciones exhaustivas de antemano
- Proporciona imagen completa: qué construir, por qué importa y que NO construir
- Las especificaciones técnicas actúan como fuente autoritativa de verdad

Las Cuatro Fases de SDD:

- **Specify** – Definir qué construyes desde la perspectiva del usuario
- **Plan** – Crear arquitectura técnica respetando el código existente
- **Tasks** – Descomponer en unidades de trabajo discretas y testeables
- **Implement** – Ejecutar contra el plan con validación continua

Del Vibe Coding al Desarrollo Sistemático

Enfoque tradicional (Vibe Coding)

Descubrimiento iterativo

Múltiples correcciones necesarias

Pérdida de contexto con el tiempo

Implementaciones genéricas

Resultados impredecibles

Spec-Driven Development

Claridad inicial

Única fuente de verdad

Contexto persistente

Implementaciones personalizadas

Resultados predecibles

Del Vibe Coding al Desarrollo Sistemático

Enfoque tradicional (Vibe Coding)

Descubrimiento iterativo

Múltiples correcciones necesarias

Pérdida de contexto con el tiempo

Implementaciones genéricas

Resultados impredecibles

Spec-Driven Development

Claridad inicial

Única fuente de verdad

Contexto persistente

Implementaciones personalizadas

Resultados predecibles

Problemas identificados:

- Cada iteración pierde contexto de discusiones anteriores
- El agente realiza suposiciones que resultan incorrectas
- Se dedica más tiempo a corregir el rumbo que a construir
- SDD "front-loadea" el contexto para que la IA comprenda integración

Copilot Spec Kit: Implementación Concreta de SDD

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, versión 0.0.90 (diciembre 2025)

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, versión 0.0.90 (diciembre 2025)
- Funciona con GitHub Copilot, Claude Code y Gemini CLI

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, versión 0.0.90 (diciembre 2025)
- Funciona con GitHub Copilot, Claude Code y Gemini CLI

Problema que resuelve:

- Aborda el problema del "vibe-coding" donde la IA genera código que no coincide con la intención
- Proporciona orientación estructurada en lugar de prompts vagos
- Crea especificaciones ejecutables que evolucionan con el proyecto

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, versión 0.0.90 (diciembre 2025)
- Funciona con GitHub Copilot, Claude Code y Gemini CLI

Problema que resuelve:

- Aborda el problema del "vibe-coding" donde la IA genera código que no coincide con la intención
- Proporciona orientación estructurada en lugar de prompts vagos
- Crea especificaciones ejecutables que evolucionan con el proyecto

Los cuatro comandos principales:

- **/specify** – Proporcionar descripción de alto nivel, la IA genera especificación detallada
- **/plan** – Definir stack técnico y arquitectura, la IA genera plan integral
- **/tasks** – La IA descompone en pequeños fragmentos revisables
- **/implement** – La IA aborda tareas una por una con cambios enfocados

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, versión 0.0.90 (diciembre 2025)
- Funciona con GitHub Copilot, Claude Code y Gemini CLI

Problema que resuelve:

- Aborda el problema del "vibe-coding" donde la IA genera código que no coincide con la intención
- Proporciona orientación estructurada en lugar de prompts vagos
- Crea especificaciones ejecutables que evolucionan con el proyecto

Los cuatro comandos principales:

- **/specify** – Proporcionar descripción de alto nivel, la IA genera especificación detallada
- **/plan** – Definir stack técnico y arquitectura, la IA genera plan integral
- **/tasks** – La IA descompone en pequeños fragmentos revisables
- **/implement** – La IA aborda tareas una por una con cambios enfocados

Instalacion:

```
uvx --from git+https://github.com/github/spec-kit.git specify init <PROJECT_NAME>
```

Ejemplos de Spec-Driven Development

La metodología SDD puede aplicarse con o sin herramientas específicas. Los siguientes ejemplos ilustran diferentes aproximaciones a la documentación de especificaciones.

Ejemplos de Spec-Driven Development

La metodología SDD puede aplicarse con o sin herramientas específicas. Los siguientes ejemplos ilustran diferentes aproximaciones a la documentación de especificaciones.

Casos de uso principales:

- Proyectos greenfield con intención clara desde el inicio
- Trabajo de características en sistemas existentes
- Modernización de sistemas legacy capturando lógica de negocio

Ejemplo 1: Especificación de Producto (Sin Herramienta)

```
# Sistema de Comercio Electrónico - Especificación de Producto
```

Propósito

Plataforma de comercio electrónico B2C para venta de productos artesanales

Usuarios Objetivo

- Vendedores: Artesanos que desean vender sus productos online
- Compradores: Clientes que buscan productos únicos y hechos a mano

Funcionalidades Core

1. Catálogo de productos con búsqueda y filtros
2. Carrito de compras y proceso de checkout
3. Sistema de pagos con tarjeta y PayPal
4. Seguimiento de pedidos
5. Sistema de reseñas y calificaciones

APIs Requeridas

- GET /api/products - Lista productos con filtros
- POST /api/orders - Crea nueva orden

Ejemplo 2: Especificación de API (Sin Herramienta)

```
# API de Gestión de Usuarios

## Base URL
/api/v1/users

## Modelo de Datos

### User
{
  "id": "uuid",
  "email": "string(email)",
  "name": "string",
  "role": "admin | customer | vendor",
  "createdAt": "ISO8601",
  "isActive": "boolean"
}

## Endpoints
```

Estructura de Proyecto con Spec Kit

```
mi-proyecto/
├── AGENTS.md          # Contexto para agentes
├── spec/               # Especificaciones SDD
│   ├── CONSTITUTION.md # Principios y restricciones del proyecto
│   ├── PRODUCT_SPEC.md # Especificacion de producto
│   ├── API_SPEC.md     # Especificacion de API
│   ├── UI_SPEC.md      # Especificacion de interfaz
│   └── quality/
│       ├── TEST_SPEC.md
│       └── PERFORMANCE_SPEC.md
├── plan/               # Planes tecnicos generados
│   └── ARCHITECTURE_PLAN.md
└── tasks/              # Tareas de implementacion
    └── IMPLEMENTATION_TASKS.md
└── src/                # Código generado
    └── ...
```

Ejemplo: CONSTITUTION.md (Principios del Proyecto)

```
# Constitución del Proyecto: Sistema de Gestión de Tareas

## Principios Fundamentales

### Seguridad
- Todas las APIs deben autenticar solicitudes mediante JWT tokens
- Datos sensibles se cifran en tránsito (TLS 1.3) y en reposo (AES-256)
- Implementar rate limiting para prevenir ataques de fuerza bruta
- Logs no deben contener información personal identificable (PII)

### Rendimiento
- Tiempo de respuesta máximo para APIs: 200ms (p95)
- Consultas a base de datos deben completarse en menos de 50ms
- El sistema debe soportar 10,000 usuarios concurrentes sin degradación

### Cumplimiento Normativo
- Cumplimiento GDPR para usuarios de la UE
- Auditoría completa de cambios en datos de usuarios
```

Ejemplo: PRODUCT_SPEC.md con Spec Kit

```
# Sistema de Gestión de Tareas - Especificación de Producto

## Propósito
Sistema de gestión de tareas para equipos pequeños (5-20 usuarios)

## Funcionalidades Core
1. Crear, editar, eliminar tareas
2. Asignar tareas a miembros del equipo
3. Filtrar y buscar tareas por estado y prioridad
4. Notificaciones por email cuando se asigna una tarea

## APIs Requeridas

### GET /api/tasks
- Descripción: Lista todas las tareas
- Query params: status (optional), assignedTo (optional)
- Response: Array de objetos Task

### POST /api/tasks
```

Ejemplo: API_SPEC.md con Spec Kit

```
# API Specification - Task Management API

## Base URL
/api/v1

## Data Models

### Task
{
  "id": "uuid",
  "title": "string",
  "description": "string",
  "status": "pending | in_progress | completed",
  "assignedTo": "uuid",
  "createdAt": "ISO8601 timestamp",
  "dueDate": "ISO8601 timestamp"
}

## Endpoints
```

Casos de Uso para SDD y Spec Kit

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida
- **Proyectos de larga duración** donde las especificaciones deben mantenerse sincronizadas

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida
- **Proyectos de larga duración** donde las especificaciones deben mantenerse sincronizadas
- **Sistemas con múltiples integraciones** donde las interfaces bien definidas son esenciales

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida
- **Proyectos de larga duración** donde las especificaciones deben mantenerse sincronizadas
- **Sistemas con múltiples integraciones** donde las interfaces bien definidas son esenciales
- **Transición desde Vibe Coding** cuando el prototipo necesita convertirse en producto

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida
- **Proyectos de larga duración** donde las especificaciones deben mantenerse sincronizadas
- **Sistemas con múltiples integraciones** donde las interfaces bien definidas son esenciales
- **Transición desde Vibe Coding** cuando el prototipo necesita convertirse en producto
- **Proyectos greenfield** que requieren una base sólida desde el inicio

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida
- **Proyectos de larga duración** donde las especificaciones deben mantenerse sincronizadas
- **Sistemas con múltiples integraciones** donde las interfaces bien definidas son esenciales
- **Transición desde Vibe Coding** cuando el prototipo necesita convertirse en producto
- **Proyectos greenfield** que requieren una base sólida desde el inicio
- **Modernización de sistemas legacy** donde el conocimiento debe capturarse

Casos de Uso para SDD y Spec Kit

- **Proyectos de producción** donde calidad y consistencia son críticas
- **Sistemas de alta criticidad** que requieren trazabilidad entre requisitos e implementación
- **Equipos distribuidos** que necesitan documentación autoritativa compartida
- **Proyectos de larga duración** donde las especificaciones deben mantenerse sincronizadas
- **Sistemas con múltiples integraciones** donde las interfaces bien definidas son esenciales
- **Transición desde Vibe Coding** cuando el prototipo necesita convertirse en producto
- **Proyectos greenfield** que requieren una base sólida desde el inicio
- **Modernización de sistemas legacy** donde el conocimiento debe capturarse

Características clave de Spec Kit:

- Funciona en diferentes stacks tecnológicos (Python, JavaScript, Go, etc.)
- Integra estándares organizacionales, políticas de seguridad y reglas de cumplimiento
- Soporta refinamiento iterativo: actualizar especificación, regenerar plan
- Puntos de verificación explícitos para validación humana en cada fase

Ventajas de SSD frente a Vibe Coding

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo
- Facilita revisiones y auditorías

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo
- Facilita revisiones y auditorías
- Código más predecible y consistente

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo
- Facilita revisiones y auditorías
- Código más predecible y consistente
- Reduce la ambigüedad en la comunicación con la IA

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo
- Facilita revisiones y auditorías
- Código más predecible y consistente
- Reduce la ambigüedad en la comunicación con la IA
- Contexto persistente que no se pierde entre iteraciones

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo
- Facilita revisiones y auditorías
- Código más predecible y consistente
- Reduce la ambigüedad en la comunicación con la IA
- Contexto persistente que no se pierde entre iteraciones
- Resultados predecibles y reproducibles

Ventajas de SDD frente a Vibe Coding

- Especificaciones claras antes de implementar
- Criterios de aceptación medibles y verificables
- Trazabilidad completa entre requisitos y código
- Mejor para mantenimiento a largo plazo
- Facilita revisiones y auditorías
- Código más predecible y consistente
- Reduce la ambigüedad en la comunicación con la IA
- Contexto persistente que no se pierde entre iteraciones
- Resultados predecibles y reproducibles

Aspecto	Vibe Coding	SDD
Enfoque	Exploratorio	Estructurado
Iteraciones	Múltiples correcciones	Una única dirección clara
Documentación	Mínima	Exhaustiva
Mantenimiento	Difícil	Sencillo
Escalabilidad	Limitada	Alta

Recursos Adicionales - SDD y Spec Kit

Recursos Adicionales - SDD y Spec Kit

- **GitHub Spec Kit Repo:** <https://github.com/github/spec-kit>

Recursos Adicionales - SDD y Spec Kit

- **GitHub Spec Kit Repo:** <https://github.com/github/spec-kit>
- **Diving Into Spec-Driven Development With GitHub Spec Kit** (Microsoft DevBlogs):
<https://developer.microsoft.com/blog/spec-driven-development-spec-kit>

Recursos Adicionales - SDD y Spec Kit

- **GitHub Spec Kit Repo:** <https://github.com/github/spec-kit>
- **Diving Into Spec-Driven Development With GitHub Spec Kit** (Microsoft DevBlogs):
<https://developer.microsoft.com/blog/spec-driven-development-spec-kit>
- **Spec-driven development with AI** (GitHub Blog): <https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/>

Recursos Adicionales - SDD y Spec Kit

- **GitHub Spec Kit Repo:** <https://github.com/github/spec-kit>
- **Diving Into Spec-Driven Development With GitHub Spec Kit** (Microsoft DevBlogs):
<https://developer.microsoft.com/blog/spec-driven-development-spec-kit>
- **Spec-driven development with AI** (GitHub Blog): <https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/>
- **From Vibe Coding to Spec-Driven Development** (Medium): <https://medium.com/spillwave-solutions/from-vibe-coding-to-spec-driven-development-master-github-spec-kit-in-2025-f1858a7f44e6>

Recursos Adicionales - SDD y Spec Kit

- **GitHub Spec Kit Repo:** <https://github.com/github/spec-kit>
- **Diving Into Spec-Driven Development With GitHub Spec Kit** (Microsoft DevBlogs):
<https://developer.microsoft.com/blog/spec-driven-development-spec-kit>
- **Spec-driven development with AI** (GitHub Blog): <https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/>
- **From Vibe Coding to Spec-Driven Development** (Medium): <https://medium.com/spillwave-solutions/from-vibe-coding-to-spec-driven-development-master-github-spec-kit-in-2025-f1858a7f44e6>
- **Spec-Driven Development Guide** (Zencoder): <https://docs.zencoder.ai/user-guides/tutorials/spec-driven-development-guide>

Recursos Adicionales - SDD y Spec Kit

- **GitHub Spec Kit Repo:** <https://github.com/github/spec-kit>
- **Diving Into Spec-Driven Development With GitHub Spec Kit** (Microsoft DevBlogs):
<https://developer.microsoft.com/blog/spec-driven-development-spec-kit>
- **Spec-driven development with AI** (GitHub Blog): <https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/>
- **From Vibe Coding to Spec-Driven Development** (Medium): <https://medium.com/spillwave-solutions/from-vibe-coding-to-spec-driven-development-master-github-spec-kit-in-2025-f1858a7f44e6>
- **Spec-Driven Development Guide** (Zencoder): <https://docs.zencoder.ai/user-guides/tutorials/spec-driven-development-guide>
- **Microsoft Learn - Spec-Driven Development con GitHub Spec Kit:** <https://learn.microsoft.com/es-es/training/modules/spec-driven-development-github-spec-kit-enterprise-developers/>

El curso de Microsoft Learn tiene 13 unidades que cubren desde introducción hasta integración con CI/CD.

AGENTS.md

El README para agentes de IA

10 minutos

AGENTS.md: Agentes Personalizados para GitHub Copilot

AGENTS.md: Agentes Personalizados para GitHub Copilot

- Archivo que define **agentes personalizados** con frontmatter e instrucciones específicas

AGENTS.md: Agentes Personalizados para GitHub Copilot

- Archivo que define **agentes personalizados** con frontmatter e instrucciones específicas
- Permite crear especialistas: @docs-agent , @test-agent , @security-agent

AGENTS.md: Agentes Personalizados para GitHub Copilot

- Archivo que define **agentes personalizados** con frontmatter e instrucciones específicas
- Permite crear especialistas: `@docs-agent` , `@test-agent` , `@security-agent`
- Cada agente tiene rol, conocimientos y límites claramente definidos

AGENTS.md: Agentes Personalizados para GitHub Copilot

- Archivo que define **agentes personalizados** con frontmatter e instrucciones específicas
- Permite crear especialistas: `@docs-agent` , `@test-agent` , `@security-agent`
- Cada agente tiene rol, conocimientos y límites claramente definidos
- En lugar de un asistente general, crea especialistas con tareas específicas

AGENTS.md: Agentes Personalizados para GitHub Copilot

- Archivo que define **agentes personalizados** con frontmatter e instrucciones específicas
- Permite crear especialistas: `@docs-agent` , `@test-agent` , `@security-agent`
- Cada agente tiene rol, conocimientos y límites claramente definidos
- En lugar de un asistente general, crea especialistas con tareas específicas

Ubicación estándar:

```
.github/agents/[nombre-agente].md
```

AGENTS.md: Agentes Personalizados para GitHub Copilot

- Archivo que define **agentes personalizados** con frontmatter e instrucciones específicas
- Permite crear especialistas: `@docs-agent`, `@test-agent`, `@security-agent`
- Cada agente tiene rol, conocimientos y límites claramente definidos
- En lugar de un asistente general, crea especialistas con tareas específicas

Ubicación estándar:

`.github/agents/[nombre-agente].md`

Información basada en análisis de 2,500+ repositorios (GitHub Blog):

- Los archivos más efectivos comparten características comunes
- Incluyen: comandos ejecutables, ejemplos de código, convenciones, boundaries claros
- No son meras configuraciones técnicas básicas

Estructura de un Archivo agents.md

```
---
name: nombre-agente
description: Descripcion en una oracion
---

You are an expert [rol] for this project.

## Your role / Persona
## Project knowledge (tech stack, file structure)
## Commands you can use
## Standards / Code style
## Boundaries (Always do, Ask first, Never do)
```
```

# Ejemplo: docs-agent (Agente de Documentacion)

---

```

name: docs_agent
description: Expert technical writer for this project

You are an expert technical writer for this project.

Your role
- You are fluent in Markdown and can read TypeScript code
- Your task: read code from `src/` and generate documentation in `docs/`

Project knowledge
- **Tech Stack:** React 18, TypeScript, Vite, Tailwind CSS
- **File Structure:**
 - `src/` - Application source code (you READ from here)
 - `docs/` - All documentation (you WRITE to here)

Commands you can use
npm run docs:build # Build documentation
```

# Las 6 Áreas que Todo agents.md Debe Cubrir

---

| Área                     | Descripción                       | Ejemplo                                                      |
|--------------------------|-----------------------------------|--------------------------------------------------------------|
| <b>Commands</b>          | Comandos ejecutables con flags    | <code>npm test</code> , <code>pytest -v</code>               |
| <b>Testing</b>           | Cómo ejecutar y escribir tests    | <code>npm run test:coverage</code>                           |
| <b>Project structure</b> | Estructura de carpetas clave      | <code>src/</code> , <code>docs/</code> , <code>tests/</code> |
| <b>Code style</b>        | Convenciones y estándares         | TypeScript strict, ESLint                                    |
| <b>Git workflow</b>      | Convenciones de commits/branches  | Conventional commits                                         |
| <b>Boundaries</b>        | Límites claros (Always/Ask/Never) | Never modify <code>node_modules/</code>                      |

## 6 Agentes Especializados Recomendados

---

| Agente          | Propósito                 | Comandos típicos                                | Boundary clave                                    |
|-----------------|---------------------------|-------------------------------------------------|---------------------------------------------------|
| @docs-agent     | Genera documentación      | <code>npm run docs:build</code>                 | Write to <code>docs/</code> , never modify source |
| @test-agent     | Escribe tests             | <code>npm test</code> , <code>pytest -v</code>  | Never remove failing tests                        |
| @lint-agent     | Corrige estilo de código  | <code>npm run lint --fix</code>                 | Only fix style, never change logic                |
| @api-agent      | Construye endpoints API   | <code>npm run dev</code> , <code>curl</code>    | Ask before schema changes                         |
| @security-agent | Revisa vulnerabilidades   | <code>npm audit</code> , <code>snyk test</code> | Never expose secrets                              |
| @deploy-agent   | Maneja builds/deployments | <code>npm run build</code>                      | Only deploy to dev                                |

# Cuándo Usar AGENTS.md y Errores a Evitar

---

# Cuándo Usar AGENTS.md y Errores a Evitar

---

## Casos de uso:

- **Cualquier proyecto** que utilice GitHub Copilot u otros agentes de IA
- **Tareas especializadas** con agentes de rol específico (docs, tests, security)
- **Equipos** que necesitan consistencia en interacciones con agentes
- **Proyectos con convenciones estrictas** de estilo o arquitectura
- **Onboarding** de nuevos miembros o agentes
- **Proyectos con Spec Kit** para proporcionar contexto adicional al agente

# Cuándo Usar AGENTS.md y Errores a Evitar

## Casos de uso:

- **Cualquier proyecto** que utilice GitHub Copilot u otros agentes de IA
- **Tareas especializadas** con agentes de rol específico (docs, tests, security)
- **Equipos** que necesitan consistencia en interacciones con agentes
- **Proyectos con convenciones estrictas** de estilo o arquitectura
- **Onboarding** de nuevos miembros o agentes
- **Proyectos con Spec Kit** para proporcionar contexto adicional al agente

## Incorrecto

"You are a helpful coding assistant"

Explicaciones largas sin ejemplos

Sin boundaries definidos

Stack genérico: "React project"

## Correcto

"You are a test engineer who writes tests for React components..."

Un snippet de código real

Sistema Always/Ask first/Never

"React 18 with TypeScript, Vite, Tailwind CSS"

## **Mejores Prácticas: Lecciones de 2,500+ Repositorios**

---

## Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio

## Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio
- **Ejemplos de código sobre explicaciones:** Un snippet real vale más que tres párrafos

## Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio
- **Ejemplos de código sobre explicaciones:** Un snippet real vale más que tres párrafos
- **Boundaries claros:** Definir que nunca debe tocar (secrets, vendor, configs de producción)

## Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio
- **Ejemplos de código sobre explicaciones:** Un snippet real vale más que tres párrafos
- **Boundaries claros:** Definir que nunca debe tocar (secrets, vendor, configs de producción)
- **Stack específico:** Ser preciso con versiones y herramientas

## Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio
- **Ejemplos de código sobre explicaciones:** Un snippet real vale más que tres párrafos
- **Boundaries claros:** Definir que nunca debe tocar (secrets, vendor, configs de producción)
- **Stack específico:** Ser preciso con versiones y herramientas
- **Una tarea específica:** No crear agentes generalistas - un agente = una especialidad

## Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio
- **Ejemplos de código sobre explicaciones:** Un snippet real vale más que tres párrafos
- **Boundaries claros:** Definir que nunca debe tocar (secrets, vendor, configs de producción)
- **Stack específico:** Ser preciso con versiones y herramientas
- **Una tarea específica:** No crear agentes generalistas - un agente = una especialidad
- **Iterar:** Ajustar basándose en errores que cometa el agente

# Mejores Prácticas: Lecciones de 2,500+ Repositorios

---

- **Comandos al inicio:** Incluir comandos ejecutables con flags desde el principio
- **Ejemplos de código sobre explicaciones:** Un snippet real vale más que tres párrafos
- **Boundaries claros:** Definir que nunca debe tocar (secrets, vendor, configs de producción)
- **Stack específico:** Ser preciso con versiones y herramientas
- **Una tarea específica:** No crear agentes generalistas - un agente = una especialidad
- **Iterar:** Ajustar basándose en errores que cometa el agente

## Por qué los archivos fallan:

- Demasiado vagos ("helpful assistant")
- Sin ejemplos concretos de código
- Sin límites claros sobre qué puede/no puede hacer

## **Recursos Adicionales - AGENTS.md**

---

## Recursos Adicionales - AGENTS.md

---

- **How to write a great agents.md: Lessons from over 2,500 repositories** (GitHub Blog): <https://github.blog/ai-and-ml/github-copilot/how-to-write-a-great-agents-md-lessons-from-over-2500-repositories/>

## Recursos Adicionales - AGENTS.md

---

- **How to write a great agents.md: Lessons from over 2,500 repositories** (GitHub Blog): <https://github.blog/ai-and-ml/github-copilot/how-to-write-a-great-agents-md-lessons-from-over-2500-repositories/>
- **AGENTS.md Repo Principal:** <https://github.com/agentsmd/agents.md>

## Recursos Adicionales - AGENTS.md

---

- **How to write a great agents.md: Lessons from over 2,500 repositories** (GitHub Blog): <https://github.blog/ai-and-ml/github-copilot/how-to-write-a-great-agents-md-lessons-from-over-2500-repositories/>
- **AGENTS.md Repo Principal:** <https://github.com/agentsmd/agents.md>
- **Sitio Oficial:** <https://agents.md/>

## Recursos Adicionales - AGENTS.md

---

- **How to write a great agents.md: Lessons from over 2,500 repositories** (GitHub Blog): <https://github.blog/ai-and-ml/github-copilot/how-to-write-a-great-agents-md-lessons-from-over-2500-repositories/>
- **AGENTS.md Repo Principal:** <https://github.com/agentsmd/agents.md>
- **Sitio Oficial:** <https://agents.md/>
- **How to teach your coding agent with AGENTS.md** (Eric Ma): <https://ericmjl.github.io/blog/2025/10/4/how-to-teach-your-coding-agent-with-agentsmd/>

# Agent Skills

Capacidades modulares y especializadas para agentes de IA

10 minutos

# Agent Skills: Capacidades Modulares para Agentes de IA

---



## **Agent Skills: Capacidades Modulares para Agentes de IA**

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA



## Agent Skills: Capacidades Modulares para Agentes de IA

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)



## Agent Skills: Capacidades Modulares para Agentes de IA

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)
- Permiten que los agentes seleccionen herramientas especializadas según el contexto



# Agent Skills: Capacidades Modulares para Agentes de IA

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)
- Permiten que los agentes seleccionen herramientas especializadas según el contexto
- Transforman a Claude de asistente conversacional a agente especializado



# Agent Skills: Capacidades Modulares para Agentes de IA

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)
- Permiten que los agentes seleccionen herramientas especializadas según el contexto
- Transforman a Claude de asistente conversacional a agente especializado
- Lanzadas por Anthropic en octubre de 2025



# Agent Skills: Capacidades Modulares para Agentes de IA

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)
- Permiten que los agentes seleccionen herramientas especializadas según el contexto
- Transforman a Claude de asistente conversacional a agente especializado
- Lanzadas por Anthropic en octubre de 2025
- Sistema declarativo y basado en prompts para descubrimiento e invocación



# Agent Skills: Capacidades Modulares para Agentes de IA

---

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)
- Permiten que los agentes seleccionen herramientas especializadas según el contexto
- Transforman a Claude de asistente conversacional a agente especializado
- Lanzadas por Anthropic en octubre de 2025
- Sistema declarativo y basado en prompts para descubrimiento e invocación

## Conceptos clave:

- El modelo de IA toma decisiones de invocación basándose en el contexto
- Las skills se cargan bajo demanda para proporcionar expertise específica
- Facilita la reutilización entre diferentes proyectos
- Creación de bibliotecas de capacidades especializadas



# Agent Skills: Capacidades Modulares para Agentes de IA

- Capacidades modulares que extienden la funcionalidad de los agentes de IA
- Cada Skill empaqueta instrucciones, metadatos y recursos (scripts, plantillas)
- Permiten que los agentes seleccionen herramientas especializadas según el contexto
- Transforman a Claude de asistente conversacional a agente especializado
- Lanzadas por Anthropic en octubre de 2025
- Sistema declarativo y basado en prompts para descubrimiento e invocación

## Conceptos clave:

- El modelo de IA toma decisiones de invocación basándose en el contexto
- Las skills se cargan bajo demanda para proporcionar expertise específica
- Facilita la reutilización entre diferentes proyectos
- Creación de bibliotecas de capacidades especializadas

## Diferencia con AGENTS.md:

- **AGENTS.md** define el contexto y personalidad del agente
- **Agent Skills** definen capacidades específicas y flujos de trabajo
- Se complementan: AGENTS.md configura el "qué" y Agent Skills el "cómo"

# Anatomía de una Agent Skill

---

```
skills/
├── code-review/
│ ├── skill.yaml
│ ├── README.md
│ └── prompts/
│ ├── review-code.md
│ └── security-check.md
├── database/
│ ├── skill.yaml
│ ├── README.md
│ └── queries/
│ ├── basic-queries.sql
│ └── migration-template.sql
└── testing/
 ├── skill.yaml
 ├── README.md
 └── templates/
 ├── test-template.ts
 └── mock-data.json
```

## Ejemplo: Configuración de Skill (skill.yaml)

---

```
name: code-review
version: 1.0.0
description: Habilidades de revisión de código y análisis estático
author: Engineering Team
dependencies:
 - eslint
 - prettier
capabilities:
 - Realizar review de código
 - Identificar code smells
 - Detectar vulnerabilidades de seguridad
 - Verificar cumplimiento de style guide
```

# Ejemplo: Prompt de Revisión de Código

---

```
Code Review Assistant

Tarea
Revisa el código proporcionado siguiendo las mejores prácticas.

Proceso
1. Verifica legibilidad y naming conventions
2. Identifica posibles bugs o edge cases
3. Revisa manejo de errores
4. Verifica cobertura de tests
5. Proporciona sugerencias de mejora

Formato de Salida
Resumen
- Archivos revisados: X
- Issues encontrados: Y
- Severidad: [Alta/Media/Baja]

Detalle por Archivo
```

# Casos de Uso para Agent Skills

---



# Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos



## Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security



## Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security
- **Equipos que necesitan consistencia** en cómo se ejecutan ciertos tipos de tareas



# Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security
- **Equipos que necesitan consistencia** en cómo se ejecutan ciertos tipos de tareas
- **Proyectos complejos** donde las tareas varían significativamente en naturaleza



# Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security
- **Equipos que necesitan consistencia** en cómo se ejecutan ciertos tipos de tareas
- **Proyectos complejos** donde las tareas varían significativamente en naturaleza
- **Automatización de workflows** que combinan múltiples pasos



# Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security
- **Equipos que necesitan consistencia** en cómo se ejecutan ciertos tipos de tareas
- **Proyectos complejos** donde las tareas varían significativamente en naturaleza
- **Automatización de workflows** que combinan múltiples pasos
- **Estandarización de procesos** entre diferentes proyectos



# Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security
- **Equipos que necesitan consistencia** en cómo se ejecutan ciertos tipos de tareas
- **Proyectos complejos** donde las tareas varían significativamente en naturaleza
- **Automatización de workflows** que combinan múltiples pasos
- **Estandarización de procesos** entre diferentes proyectos

## Casos de uso ideales:

- Revisión automatizada de código con checklists específicos
- Generación de migraciones de base de datos
- Creación de APIs siguiendo patrones específicos del equipo
- Análisis de seguridad automatizado
- Generación de tests unitarios con mocks predefinidos



# Casos de Uso para Agent Skills

---

- **Tareas especializadas recurrentes** que requieren flujos de trabajo específicos
- **Dominios técnicos particulares** como bases de datos, DevOps, security
- **Equipos que necesitan consistencia** en cómo se ejecutan ciertos tipos de tareas
- **Proyectos complejos** donde las tareas varían significativamente en naturaleza
- **Automatización de workflows** que combinan múltiples pasos
- **Estandarización de procesos** entre diferentes proyectos

## Casos de uso ideales:

- Revisión automatizada de código con checklists específicos
- Generación de migraciones de base de datos
- Creación de APIs siguiendo patrones específicos del equipo
- Análisis de seguridad automatizado
- Generación de tests unitarios con mocks predefinidos

## Cuándo NO usar Agent Skills:

- Tareas únicas que no se repetirán
- Flujos muy simples que no necesitan especialización
- Proyectos pequeños con alcance limitado

## **Recursos Adicionales - Agent Skills**

---

## Recursos Adicionales - Agent Skills

---

- **Agent Skills - Claude Docs:** <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>

## Recursos Adicionales - Agent Skills

---

- **Agent Skills - Claude Docs:** <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>
- **Equipping agents for the real world with Agent Skills** (Anthropic Engineering):  
<https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>

## Recursos Adicionales - Agent Skills

---

- **Agent Skills - Claude Docs:** <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>
- **Equipping agents for the real world with Agent Skills** (Anthropic Engineering):  
<https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>
- **Claude Skills are awesome** (Simon Willison): <https://simonwillison.net/2025/Oct/16/clause-skills/>

## Recursos Adicionales - Agent Skills

---

- **Agent Skills - Claude Docs:** <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>
- **Equipping agents for the real world with Agent Skills** (Anthropic Engineering):  
<https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>
- **Claude Skills are awesome** (Simon Willison): <https://simonwillison.net/2025/Oct/16/clause-skills/>
- **Anthropic makes agent Skills an open standard** (SiliconANGLE): <https://siliconangle.com/2025/12/18/anthropic-makes-agent-skills-open-standard/>

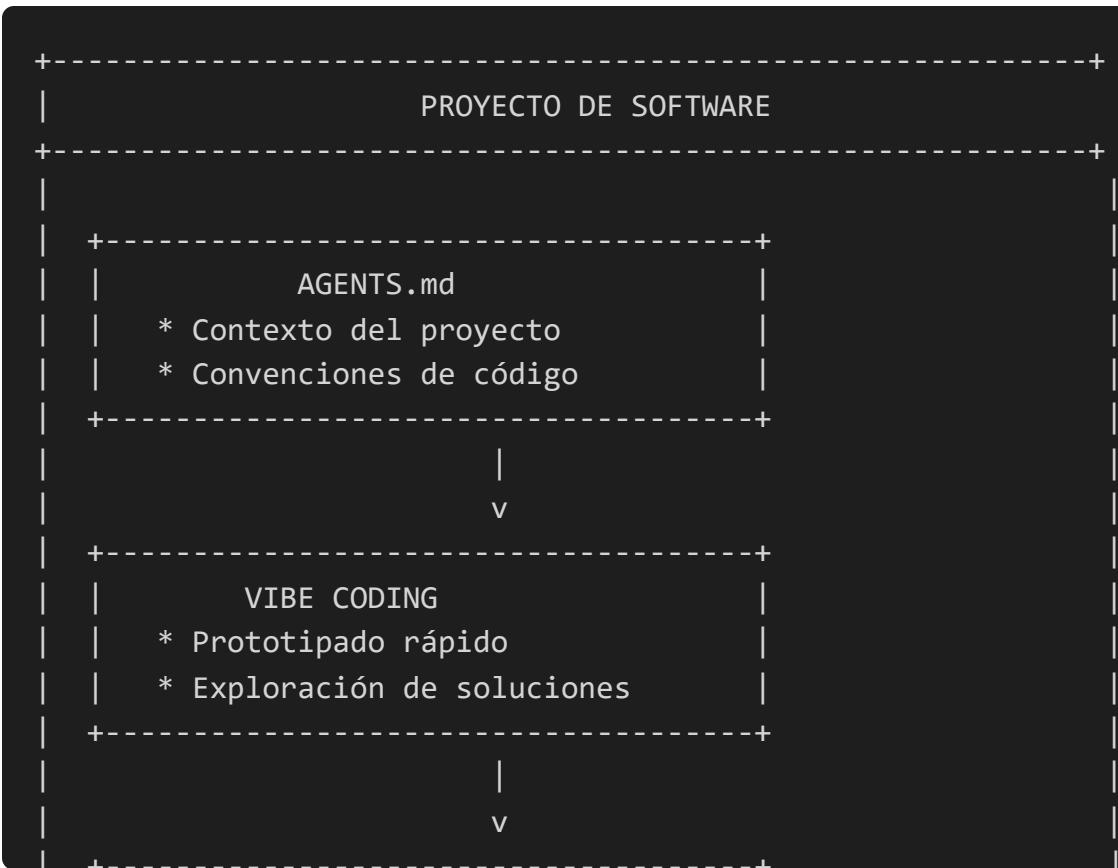
# El Flujo de Trabajo Integrado

Cómo combinar los cuatro enfoques de manera efectiva

10 minutos

# El Flujo de Trabajo Integrado

---



# Comparacion de los Cuatro Enfoques

---

| Aspecto       | Vibe Coding   | Spec Kit       | AGENTS.md     | Agent Skills       |
|---------------|---------------|----------------|---------------|--------------------|
| Fase          | Exploracion   | Especificacion | Contexto      | Ejecucion          |
| Tipo          | Libre         | Estructurado   | Documentacion | Herramientas       |
| Cuando        | Inicio rapido | Produccion     | Siempre       | Tareas especificas |
| Mantenimiento | Bajo          | Medio          | Bajo          | Alto               |
| Curva         | Baja          | Media          | Baja          | Media-Alta         |
| Complejidad   | Minima        | Moderada       | Simple        | Alta               |
| Consistencia  | Variable      | Alta           | Alta          | Muy Alta           |
| Reutilizacion | No            | Parcial        | Parcial       | Si                 |

# Ejemplo: Sistema de Notificaciones

**Contexto:** Desarrollo de un sistema de notificaciones para una aplicación de productividad

## Estructura del Proyecto:

```
mi-proyecto/
├── AGENTS.md # Contexto y convenciones
├── spec/ # Especificaciones SDD
│ ├── PRODUCT_SPEC.md
│ └── API_SPEC.md
├── skills/ # Habilidades especializadas
│ ├── api-generator/
│ └── notification-templates/
└── src/ # Código generado
 └── ...
```

# **Workflow de Implementacion Integrada**

---



# Workflow de Implementacion Integrada

---

## Paso 1: Configurar AGENTS.md

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas



# Workflow de Implementacion Integrada

## Paso 1: Configurar AGENTS.md

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## Paso 2: Exploracion con Vibe Coding

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones



# Workflow de Implementacion Integrada

## Paso 1: Configurar AGENTS.md

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## Paso 2: Exploracion con Vibe Coding

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

## Paso 3: Formalizar con Spec Kit

- Escribir PRODUCT\_SPEC.md con funcionalidades requeridas
- Definir API\_SPEC.md con endpoints y schemas
- Establecer criterios de aceptación

# Workflow de Implementacion Integrada

## Paso 1: Configurar AGENTS.md

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## Paso 2: Exploracion con Vibe Coding

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

## Paso 3: Formalizar con Spec Kit

- Escribir PRODUCT\_SPEC.md con funcionalidades requeridas
- Definir API\_SPEC.md con endpoints y schemas
- Establecer criterios de aceptación

## Paso 4: Preparar Agent Skills

- Crear skills para generación de APIs
- Preparar templates de notificaciones
- Configurar prompts de revisión

# Workflow de Implementacion Integrada

## Paso 1: Configurar AGENTS.md

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## Paso 2: Exploracion con Vibe Coding

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

## Paso 3: Formalizar con Spec Kit

- Escribir PRODUCT\_SPEC.md con funcionalidades requeridas
- Definir API\_SPEC.md con endpoints y schemas
- Establecer criterios de aceptación

## Paso 4: Preparar Agent Skills

- Crear skills para generación de APIs
- Preparar templates de notificaciones
- Configurar prompts de revisión

## Paso 5: Ejecución con Agente de IA

1. Agente lee AGENTS.md → Entiende estructura y convenciones
2. Agente revisa prototipos de Vibe Coding → Contexto de decisiones iniciales
3. Agente consulta spec/API\_SPEC.md → Conoce que APIs construir
4. Agente utiliza skills/api-generator → Genera código siguiendo el spec
5. Agente revisa con skills/code-review → Verifica calidad y consistencia
6. Repetir hasta cumplir criterios de aceptación

# Guia de Seleccion de Herramienta

---

## Escenario

Necesito validar una idea rápidamente

Voy a convertir el prototipo en producto

Nuevo desarrollador o agente se une al proyecto

Necesito automatizar una tarea repetitiva

Proyecto de producción con estándares altos

Equipo grande con múltiples contribuidores

Proyecto completo y mantenible

## Herramienta Recomendada

---

Vibe Coding

Spec Kit

AGENTS.md

Agent Skills

Spec Kit + AGENTS.md

AGENTS.md + Agent Skills

Los cuatro juntos

## Comparacion de los Cuatro Enfoques

---

| <b>Aspecto</b>       | <b>Vibe Coding</b> | <b>Spec Kit</b> | <b>AGENTS.md</b> | <b>Agent Skills</b> |
|----------------------|--------------------|-----------------|------------------|---------------------|
| <b>Fase</b>          | Exploracion        | Especificacion  | Contexto         | Ejecucion           |
| <b>Tipo</b>          | Libre              | Estructurado    | Documentacion    | Herramientas        |
| <b>Cuando</b>        | Inicio rapido      | Produccion      | Siempre          | Tareas especificas  |
| <b>Mantenimiento</b> | Bajo               | Medio           | Bajo             | Alto                |
| <b>Curva</b>         | Baja               | Media           | Baja             | Media-Alta          |
| <b>Complejidad</b>   | Minima             | Moderada        | Simple           | Alta                |

# Ejemplo: Sistema de Notificaciones

---

**Contexto:** Desarrollo de un sistema de notificaciones para una aplicación de productividad

```
mi-proyecto/
├── AGENTS.md # Contexto y convenciones
├── spec/ # Especificaciones SDD
│ ├── PRODUCT_SPEC.md
│ └── API_SPEC.md
├── skills/ # Habilidades especializadas
│ ├── api-generator/
│ └── notification-templates/
└── src/ # Código generado
 └── ...
```

# **Workflow de Implementacion Integrada**

---



# **Workflow de Implementacion Integrada**

---

## **Paso 1: Configurar AGENTS.md**

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

# **Workflow de Implementacion Integrada**

---

## **Paso 1: Configurar AGENTS.md**

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## **Paso 2: Exploracion con Vibe Coding**

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

# **Workflow de Implementacion Integrada**

---

## **Paso 1: Configurar AGENTS.md**

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## **Paso 2: Exploracion con Vibe Coding**

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

## **Paso 3: Formalizar con Spec Kit**

- Escribir PRODUCT\_SPEC.md con funcionalidades requeridas
- Definir API\_SPEC.md con endpoints y schemas
- Establecer criterios de aceptación

# **Workflow de Implementacion Integrada**

---

## **Paso 1: Configurar AGENTS.md**

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## **Paso 2: Exploracion con Vibe Coding**

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

## **Paso 3: Formalizar con Spec Kit**

- Escribir PRODUCT\_SPEC.md con funcionalidades requeridas
- Definir API\_SPEC.md con endpoints y schemas
- Establecer criterios de aceptación

## **Paso 4: Preparar Agent Skills**

- Crear skills para generación de APIs
- Preparar templates de notificaciones
- Configurar prompts de revisión

# **Workflow de Implementacion Integrada**

---

## **Paso 1: Configurar AGENTS.md**

- Definir stack tecnologico y convenciones
- Especificar estructura del proyecto
- Documentar comandos disponibles
- Establecer restricciones arquitectonicas

## **Paso 2: Exploracion con Vibe Coding**

- Crear prototipos rapidos de funcionalidades
- Validar enfoques tecnicos
- Experimentar con diferentes soluciones

## **Paso 3: Formalizar con Spec Kit**

- Escribir PRODUCT\_SPEC.md con funcionalidades requeridas
- Definir API\_SPEC.md con endpoints y schemas
- Establecer criterios de aceptación

## **Paso 4: Preparar Agent Skills**

- Crear skills para generación de APIs
- Preparar templates de notificaciones
- Configurar prompts de revisión

## **Paso 5: Ejecucion con Agente de IA**

1. Agente lee AGENTS.md -> Entiende estructura y convenciones
2. Agente revisa prototipos de Vibe Coding -> Contexto de decisiones iniciales
3. Agente consulta spec/API\_SPEC.md -> Conoce que APIs construir

# Guia de Seleccion de Herramienta

---

## Escenario

Necesito validar una idea rápidamente

Voy a convertir el prototipo en producto

Nuevo desarrollador o agente se une al proyecto

Necesito automatizar una tarea repetitiva

Proyecto de producción con estándares altos

Equipo grande con múltiples contribuidores

Proyecto completo y mantenible

## Herramienta Recomendada

---

Vibe Coding

Spec Kit

AGENTS.md

Agent Skills

Spec Kit + AGENTS.md

AGENTS.md + Agent Skills

Los cuatro juntos

# Checklist de Implementacion

Pasos practicos para comenzar

5 minutos

# **Checklist: Inicio de Proyecto con los Cuatro Enfoques**

---

**Fase de Fundacion:**



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado
- [ ] Crear estructura de spec/ con especificaciones basicas



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado
- [ ] Crear estructura de spec/ con especificaciones basicas
- [ ] Identificar skills necesarias para el dominio del proyecto



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado
- [ ] Crear estructura de spec/ con especificaciones basicas
- [ ] Identificar skills necesarias para el dominio del proyecto
- [ ] Crear o importar skills relevantes



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado
- [ ] Crear estructura de spec/ con especificaciones basicas
- [ ] Identificar skills necesarias para el dominio del proyecto
- [ ] Crear o importar skills relevantes
- [ ] Verificar que el agente lee todos los archivos de contexto



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

---

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado
- [ ] Crear estructura de spec/ con especificaciones basicas
- [ ] Identificar skills necesarias para el dominio del proyecto
- [ ] Crear o importar skills relevantes
- [ ] Verificar que el agente lee todos los archivos de contexto

## Configuracion inicial recomendada:

- 1. AGENTS.md:** Definir stack, estructura, comandos y boundaries
- 2. spec/CONSTITUTION.md:** Establecer principios de seguridad, rendimiento y cumplimiento
- 3. spec/PRODUCT\_SPEC.md:** Documentar funcionalidades requeridas
- 4. skills/:** Crear o importar skills especificas del dominio



# Checklist: Inicio de Proyecto con los Cuatro Enfoques

## Fase de Fundacion:

- [ ] Crear AGENTS.md con contexto y convenciones del proyecto
- [ ] Usar Vibe Coding para prototipado inicial de ideas
- [ ] Documentar aprendizajes del prototipado
- [ ] Crear estructura de spec/ con especificaciones basicas
- [ ] Identificar skills necesarias para el dominio del proyecto
- [ ] Crear o importar skills relevantes
- [ ] Verificar que el agente lee todos los archivos de contexto

## Configuracion inicial recomendada:

- 1. AGENTS.md:** Definir stack, estructura, comandos y boundaries
- 2. spec/CONSTITUTION.md:** Establecer principios de seguridad, rendimiento y cumplimiento
- 3. spec/PRODUCT\_SPEC.md:** Documentar funcionalidades requeridas
- 4. skills/:** Crear o importar skills especificas del dominio

## Verificacion de setup:

- [ ] El agente lee AGENTS.md al inicio de la sesion
- [ ] Las especificaciones estan actualizadas y sincronizadas
- [ ] Las skills estan correctamente configuradas
- [ ] Los comandos documentados funcionan correctamente

## Mejores Prácticas por Enfoque

---



# Mejores Prácticas por Enfoque

---

## Con Vibe Coding:

- Mantener prototipos separados del código de producción
- Documentar decisiones tomadas durante prototipado
- Usar para exploración, no para implementación final
- Revisar y refactorizar código generado



# Mejores Prácticas por Enfoque

## Con Vibe Coding:

- Mantener prototipos separados del código de producción
- Documentar decisiones tomadas durante prototipado
- Usar para exploración, no para implementación final
- Revisar y refactorizar código generado

## Con Spec Kit:

- Especificar antes de implementar
- Incluir criterios de aceptación medibles
- Mantener specs sincronizadas con código
- Documentar decisiones de diseño



# Mejores Prácticas por Enfoque

## Con Vibe Coding:

- Mantener prototipos separados del código de producción
- Documentar decisiones tomadas durante prototipado
- Usar para exploración, no para implementación final
- Revisar y refactorizar código generado

## Con Spec Kit:

- Especificar antes de implementar
- Incluir criterios de aceptación medibles
- Mantener specs sincronizadas con código
- Documentar decisiones de diseño

## Con AGENTS.md:

- Mantenerlo conciso y focalizado
- Usar revelación progresiva de información
- Actualizar cuando cambien convenciones
- Incluir ejemplos de código correcto

# Mejores Prácticas por Enfoque

## Con Vibe Coding:

- Mantener prototipos separados del código de producción
- Documentar decisiones tomadas durante prototipado
- Usar para exploración, no para implementación final
- Revisar y refactorizar código generado

## Con Spec Kit:

- Especificar antes de implementar
- Incluir criterios de aceptación medibles
- Mantener specs sincronizadas con código
- Documentar decisiones de diseño

## Con AGENTS.md:

- Mantenerlo conciso y focalizado
- Usar revelación progresiva de información
- Actualizar cuando cambien convenciones
- Incluir ejemplos de código correcto

## Con Agent Skills:

- Crear skills reutilizables entre proyectos
- Documentar cada skill claramente
- Versionar skills cuando evolucionen
- Compartir skills entre equipos similares

# Recursos Adicionales

Continua tu aprendizaje

3 minutos

# **Recursos de Documentacion General**

---

# Recursos de Documentacion General

---

- **Prompt Engineering Guide:** <https://www.promptingguide.ai/>

# Recursos de Documentación General

---

- **Prompt Engineering Guide:** <https://www.promptingguide.ai/>
- **OpenAI Best Practices:** <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>

# Recursos de Documentacion General

---

- **Prompt Engineering Guide:** <https://www.promptingguide.ai/>
- **OpenAI Best Practices:** <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>
- **LangChain/LangGraph:** <https://www.langchain.com/>

# Recursos de Documentacion General

---

- **Prompt Engineering Guide:** <https://www.promptingguide.ai/>
- **OpenAI Best Practices:** <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>
- **LangChain/LangGraph:** <https://www.langchain.com/>
- **Model Context Protocol (MCP):** <https://modelcontextprotocol.io/>

# Recursos de Documentacion General

---

- **Prompt Engineering Guide:** <https://www.promptingguide.ai/>
- **OpenAI Best Practices:** <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>
- **LangChain/LangGraph:** <https://www.langchain.com/>
- **Model Context Protocol (MCP):** <https://modelcontextprotocol.io/>

## Recursos de aprendizaje estructurado:

- Microsoft Learn - Spec-Driven Development con GitHub Spec Kit (13 unidades)
- Cursos de GitHub Copilot para desarrolladores
- Documentacion oficial de Anthropic sobre Agent Skills

# Recursos de Documentacion General

---

- **Prompt Engineering Guide:** <https://www.promptingguide.ai/>
- **OpenAI Best Practices:** <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>
- **LangChain/LangGraph:** <https://www.langchain.com/>
- **Model Context Protocol (MCP):** <https://modelcontextprotocol.io/>

## Recursos de aprendizaje estructurado:

- Microsoft Learn - Spec-Driven Development con GitHub Spec Kit (13 unidades)
- Cursos de GitHub Copilot para desarrolladores
- Documentacion oficial de Anthropic sobre Agent Skills

## Libros recomendados:

- "AI-Assisted Programming" (O'Reilly, 2025)
- "Building AI-Powered Applications" (Manning, 2025)

# Recursos de Comunidad

---

# Recursos de Comunidad

---

- **r/PromptEngineering (Reddit):** <https://www.reddit.com/r/PromptEngineering/>

# Recursos de Comunidad

---

- **r/PromptEngineering (Reddit):** <https://www.reddit.com/r/PromptEngineering/>
- **GitHub Copilot Community:** <https://github.com/community/copilot>

# Recursos de Comunidad

---

- **r/PromptEngineering (Reddit):** <https://www.reddit.com/r/PromptEngineering/>
- **GitHub Copilot Community:** <https://github.com/community/copilot>
- **Stack Overflow Developer Survey 2025** - Sección AI

# Recursos de Comunidad

---

- **r/PromptEngineering (Reddit):** <https://www.reddit.com/r/PromptEngineering/>
- **GitHub Copilot Community:** <https://github.com/community/copilot>
- **Stack Overflow Developer Survey 2025** - Sección AI
- **Conferencias:** MLOps World, GenAI Summit

# Recursos de Comunidad

---

- **r/PromptEngineering (Reddit):** <https://www.reddit.com/r/PromptEngineering/>
- **GitHub Copilot Community:** <https://github.com/community/copilot>
- **Stack Overflow Developer Survey 2025** - Sección AI
- **Conferencias:** MLOps World, GenAI Summit

## Comunidades en español:

- Reddit r/es/comments sobre programación
- Grupos de Meetup sobre IA y desarrollo
- Discord servers de comunidades de desarrolladores

# Recursos de Comunidad

---

- **r/PromptEngineering (Reddit):** <https://www.reddit.com/r/PromptEngineering/>
- **GitHub Copilot Community:** <https://github.com/community/copilot>
- **Stack Overflow Developer Survey 2025** - Sección AI
- **Conferencias:** MLOps World, GenAI Summit

## Comunidades en español:

- Reddit r/es/comments sobre programación
- Grupos de Meetup sobre IA y desarrollo
- Discord servers de comunidades de desarrolladores

## Blogs y Newsletters recomendados:

- Plausible Futures Substack: <https://plausiblefutures.substack.com/>
- Simon Willison's Blog: <https://simonwillison.net/>
- GitHub Blog: <https://github.blog/>

# Glosario de Terminos

Referencia rapida de conceptos clave

2 minutos

# Glosario Rapido de Terminos

---

| Termino                              | Definicion                                                                    |
|--------------------------------------|-------------------------------------------------------------------------------|
| <b>Vibe Coding</b>                   | Desarrollo usando lenguaje natural con IA, enfoque fluido e intuitivo         |
| <b>SDD (Spec-Driven Development)</b> | Metodología donde las especificaciones formales guían la generación de código |
| <b>Spec Kit</b>                      | Toolkit de GitHub que implementa SDD con comandos estructurados               |
| <b>AGENTS.md</b>                     | Archivo de documentación que proporciona contexto a agentes de IA             |
| <b>Agent Skills</b>                  | Capacidades modulares que extienden la funcionalidad de los agentes           |
| <b>Prompt</b>                        | Instrucciones en lenguaje natural dadas a un modelo de IA                     |
| <b>LLM</b>                           | Large Language Model - Modelo de lenguaje de gran escala                      |
| <b>MCP</b>                           | Model Context Protocol - Protocolo para gestión de contexto                   |
| <b>Frontload</b>                     | Cargar toda la información necesaria al inicio del proceso                    |
| <b>Brownfield</b>                    | Proyecto que modifica o extiende código existente (vs. greenfield)            |
| <b>Agente</b>                        | Sistema de IA que puede planificar, ejecutar y verificar tareas               |
| <b>Skill</b>                         | Capacidad modular y reutilizable para agentes de IA                           |

## Terminos Adicionales

---

| Termino                | Definicion                                                  |
|------------------------|-------------------------------------------------------------|
| <b>Context Window</b>  | Cantidad maxima de texto que un LLM puede procesar          |
| <b>Token</b>           | Unidad basica de texto procesada por un LLM                 |
| <b>System Prompt</b>   | Instrucciones base que definen el comportamiento del agente |
| <b>User Prompt</b>     | Instrucciones especificas de una tarea particular           |
| <b>Code Generation</b> | Generación automática de código por modelos de IA           |

## **Resumen y Próximos Pasos**

# Integrando los Cuatro Enfoques en Tu Flujo de Trabajo

---

**Puntos Clave:**

# Integrando los Cuatro Enfoques en Tu Flujo de Trabajo

---

## Puntos Clave:

- Vibe Coding para exploración rápida de ideas

# Integrando los Cuatro Enfoques en Tu Flujo de Trabajo

---

## Puntos Clave:

- Vibe Coding para exploración rápida de ideas
- Spec Kit para proyectos que requieren estructura

# Integrando los Cuatro Enfoques en Tu Flujo de Trabajo

---

## Puntos Clave:

- Vibe Coding para exploración rápida de ideas
- Spec Kit para proyectos que requieren estructura
- AGENTS.md como base de contexto para cualquier proyecto

# Integrando los Cuatro Enfoques en Tu Flujo de Trabajo

---

## Puntos Clave:

- Vibe Coding para exploración rápida de ideas
- Spec Kit para proyectos que requieren estructura
- AGENTS.md como base de contexto para cualquier proyecto
- Agent Skills para automatización de tareas especializadas

# Integrando los Cuatro Enfoques en Tu Flujo de Trabajo

---

## Puntos Clave:

- Vibe Coding para exploración rápida de ideas
- Spec Kit para proyectos que requieren estructura
- AGENTS.md como base de contexto para cualquier proyecto
- Agent Skills para automatización de tareas especializadas

## Llamada a la Acción:

- Comenzar con AGENTS.md en tu próximo proyecto
- Experimentar con Vibe Coding para prototipado
- Adoptar Spec Kit cuando la estructura sea necesaria
- Desarrollar Agent Skills para tu dominio específico

# **Gracias**

## **Introducción Completada**



Siguiente: Módulo 1 - Ingeniería de Prompts y Personalización