netmind | a bts company

Simulación Scrum - SDD

PortalEmpleo API - Planificación de Iteraciones

Spec-Driven Development con Scrum

Sprint 0 - Foundations

Configurar entorno SDD

- spec-kit init para inicializar el proyecto
- Verificar instalación de .NET 8.0 SDK
- Configurar extensiones de Copilot en VS Code
- Verificar configuración de Git

Generar constitution.md

- Ejecutar /speckit.constitution
- Definir estándares tecnológicos (.NET 8, Clean Architecture)
- Establecer requisitos de seguridad (JWT, BCrypt)
- Documentar estándares de codificación

Crear AGENTS.md

- Definir agentes especializados (@backend-dev, @data-dev, @test-dev)
- Establecer stack tecnológico y convenciones
- Configurar comandos ejecutables
- Definir límites (Always/Ask/Never)

Documentar ADRs

- ADR-001: Clean Architecture con 4 capas
- ADR-002: Entity Framework Core con In-Memory
- ADR-003: JWT HS256 para autenticación
- ADR-004: BCrypt work factor 12 para passwords

Generar diagramas de arquitectura

- Diagrama de arquitectura general (capas)
- Diagrama de flujo de autenticación
- Diagrama de ciclo de ofertas (borrador → publicada → pausada → cerrada)
- Diagrama de flujo de postulaciones

Review: Validar artefactos fundacionales

- Verificar constitution.md con todos los estándares
- Revisar AGENTS.md con agentes especializados
- Confirmar ADRs documentados
- Validar diagramas de arquitectura

Sprint 1 - Especificación Auth

Generar spec.md para Autenticación

- RF-001: Registro de usuarios (email único, password compleja, edad ≥16)
- RF-002: Login (JWT 60min/7days, BCrypt, rate limiting)
- RF-003: Refresh token
- RF-004: GET /users/me (perfil propio)
- RF-005: PUT /users/me (update de perfil)
- RF-006: DELETE /users/me (soft delete)

Clarificación y aprobación de especificación

- Ejecutar /speckit.clarify para identificar gaps
- Responder Open Questions
- Completar checklist de aprobación (funcional/técnica/SDD)
- Actualizar spec.md con respuestas

Review: Demo de spec.md aprobado

- Presentar spec.md para Autenticación
- Verificar que todos los RFs están cubiertos
- Confirmar criterios de aceptación medibles
- Aprobación formal del Product Owner

Sprint 2 - Especificación Jobs + Plan Auth

Generar spec.md para Ofertas

- RF-007: Crear oferta (BORRADOR, title max 200, contract types)
- RF-008: Publicar oferta (BORRADOR → PUBLICADA)
- RF-009: Listar ofertas (filtros q, location, contractType, salary)
- RF-010: GET /{id} (detalle de oferta)
- RF-011: PUT /{id} (editar oferta)
- RF-012: Estados de oferta (BORRADOR/PUBLICADA/PAUSADA/CERRADA)

Generar Plan.md para Autenticación

- Estructura de proyectos (.NET 8, Clean Architecture)
- Dependencias NuGet (EF Core, JWT, BCrypt, FluentValidation)
- Fases de implementación (Domain → Infrastructure → Application → API)
- Dependencias entre tareas

Generar Tasks.md para Auth

- T001: Crear entidades de dominio (User, UserRole)
- T002: Implementar interfaces de repositorio
- T003: Crear UserRepository
- T004: Implementar AuthService
- T005: Crear AuthController
- T006: Generar tests unitarios

Review: Demo de spec.md Jobs y Plan Auth

- Presentar spec.md para Ofertas de Empleo
- Mostrar Plan.md para Autenticación
- Demostrar Tasks.md con desglose de tareas
- Aprobación de planificación

Sprint 3 - Plan Jobs + Implementación Auth

Generar Plan.md para Ofertas

- Estructura de proyectos para módulo de ofertas
- Entidades: JobOffer, JobOfferStatus, ContractType
- Servicios: JobOfferService con CRUD y transiciones de estado
- Controladores: endpoints públicos y privados

Implementar entidades y repositorios (Auth)

- User.cs: Id, Email, PasswordHash, Name, BirthDate, Phone, Role, IsActive
- UserRole enum: CANDIDATE, COMPANY, ADMIN
- IUserRepository interface
- UserRepository implementación

Generar controladores y servicios (Auth)

- AuthController: POST /auth/register, /auth/login, /auth/refresh
- UsersController: GET /users/me, PUT /users/me, DELETE /users/me
- AuthService: RegisterAsync, LoginAsync, RefreshTokenAsync
- DTOs con FluentValidation

Verificación de compilación

- dotnet build --no-restore
- Corregir errores de compilación
- Ejecutar tests unitarios
- Verificar cobertura ≥80%

Review: Demo endpoints Auth

- Demostrar registro de usuario
- Mostrar login con generación de JWT
- Probar refresh token
- Verificar perfiles y soft delete

Sprint 4 - Implementación Jobs

Implementar entidades y repositorios (Jobs)

- JobOffer.cs: Id, Title, Description, Location, Salary, ContractType, Status
- JobOfferStatus enum: BORRADOR, PUBLICADA, PAUSADA, CERRADA
- ContractType enum: TIEMPO_COMPLETO, MEDIO_TIEMPO, TEMPORAL, FREELANCE
- IJobOfferRepository, JobOfferRepository

Generar controladores Jobs

- GET /job-offers (público con filtros)
- POST /job-offers (COMPANY)
- GET /{id} (público)
- PUT /{id} (COMPANY own)
- PATCH /{id}/publish, /pause, /close (COMPANY own)

Implementar servicios de ofertas

- CreateAsync: validación y creación en BORRADOR
- PublishAsync: BORRADOR → PUBLICADA (irreversible)
- ListAsync: filtros, paginación (default 10, max 100)
- UpdateAsync: solo BORRADOR o PAUSADA
- Pause/Close: transiciones de estado

Generar eventos y handlers

- JobOfferPublishedEvent
- ApplicationSubmittedEvent
- Event handlers para notificaciones
- Observer Pattern implementado

Review: Demo endpoints Jobs

- Crear oferta en borrador
- Publicar oferta
- Listar con filtros
- Probar transiciones de estado

Sprint 5 - Implementación Apps + Testing

Implementar entidades, repositorios y servicios (Apps)

- Application.cs: Id, JobOfferId, CandidateId, Status, AppliedAt, Notes
- ApplicationStatus enum: PENDIENTE, EN_REVISION, ENTREVISTADO, ACEPTADO, RECHAZADO
- IApplicationRepository, ApplicationRepository
- ApplicationService: ApplyAsync, GetMyApplicationsAsync, UpdateStatusAsync

Generar Tests Unitarios

- AuthServiceTests: Register validación email único, Login credenciales
- UserServiceTests: Update perfil, Soft delete
- JobOfferServiceTests: Publicar oferta, Listar con filtros
- ApplicationServiceTests: Postulación única, Cambio de estado

Generar Tests de Integración

- AuthControllerTests: Register, Login, Refresh token
- JobOffersControllerTests: CRUD ofertas, filtros, paginación
- ApplicationsControllerTests: Postulación, estados
- Framework: xUnit + Moq + FluentAssertions

Verificar cobertura ≥80%

- dotnet test --collect:"XPlat Code Coverage"
- Analizarレポート de cobertura
- Añadir tests para coverage gaps
- Validar coverage mínimo alcanzado

Review: Demo con tests passing

- Ejecutar suite completa de tests
- Mostrar coverage report
- Demostrar funcionalidades con tests passing
- Aprobación de Quality Gate

Sprint 6 - Refactorización y Cierre

Análisis de code smells

- Long Method (métodos >30 líneas)
- Large Class (clases >500 líneas)
- Feature Envy
- Data Clumps
- Primitive Obsession

Ejecutar refactorización

- Extract Method para métodos largos
- Replace Primitive with Object
- Move Method a clases apropiadas
- Introduce Parameter Object

Validación final de tests

- Todos los tests pasan después de refactorizar
- Coverage se mantiene ≥80%
- Verificar funcionalidad no afectada
- Documentar refactorizaciones realizadas

Demo Final completa

- Demostrar PortalEmpleo API completo
- Flujo completo de usuario (registro → login → buscar ofertas → postularse)
- Flujo de empresa (crear oferta → publicar → gestionar postulaciones)
- Verificación vs todos los spec.md

Retrospective del curso

- ¿Qué funcionó bien con SDD + Scrum?
- ¿Qué mejoraríamos en próximos proyectos?
- Lecciones aprendidas sobre agentes IA
- Propuestas de mejora a constitution.md

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

- **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 autonomos - 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 autonomos - 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	Descripcion	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	Descripcion	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?

- 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
- 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

```
const LoginScreen = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [emailError, setEmailError] = useState('');

  const validateEmail = (email) => {
    const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return re.test(email);
  };

  const handleLogin = () => {
    if (!validateEmail(email)) {
      setEmailError('Por favor ingresa un email válido');
      return;
    }
    // Logica de login...
  };

  return (
```

Casos de Uso Ideales para Vibe Coding

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

- 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

- **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://plausiblefutures.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

- 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)	Spec-Driven Development
Descubrimiento iterativo	Claridad inicial
Múltiples correcciones necesarias	Única fuente de verdad
Pérdida de contexto con el tiempo	Contexto persistente
Implementaciones genéricas	Implementaciones personalizadas
Resultados impredecibles	Resultados predecibles

Del Vibe Coding al Desarrollo Sistemático

Enfoque tradicional (Vibe Coding)	Spec-Driven Development
Descubrimiento iterativo	Claridad inicial
Múltiples correcciones necesarias	Única fuente de verdad
Pérdida de contexto con el tiempo	Contexto persistente
Implementaciones genéricas	Implementaciones personalizadas
Resultados impredecibles	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

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

Instalacion:

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

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, version 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

Instalacion:

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

Copilot Spec Kit: Implementación Concreta de SDD

- Toolkit de código abierto desarrollado por GitHub
- Lanzado en septiembre de 2024, version 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 Gestion 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
- Descripcion: 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

- **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 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

- **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 introduccion hasta integracion con CI/CD.

AGENTS.md

El README para agentes de IA

10 minutos

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

6 Agentes Especializados Recomendados

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

Cuándo Usar AGENTS.md y Errores a Evitar

Incorrecto	Correcto
"You are a helpful coding assistant"	"You are a test engineer who writes tests for React components..."
Explicaciones largas sin ejemplos	Un snippet de código real
Sin boundaries definidos	Sistema Always/Ask first/Never
Stack genérico: "React project"	"React 18 with TypeScript, Vite, Tailwind CSS"

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	Correcto
"You are a helpful coding assistant"	"You are a test engineer who writes tests for React components..."
Explicaciones largas sin ejemplos	Un snippet de código real
Sin boundaries definidos	Sistema Always/Ask first/Never
Stack genérico: "React project"	"React 18 with TypeScript, Vite, Tailwind CSS"

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

- **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

- 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"

Anatomia 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)

```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

- **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

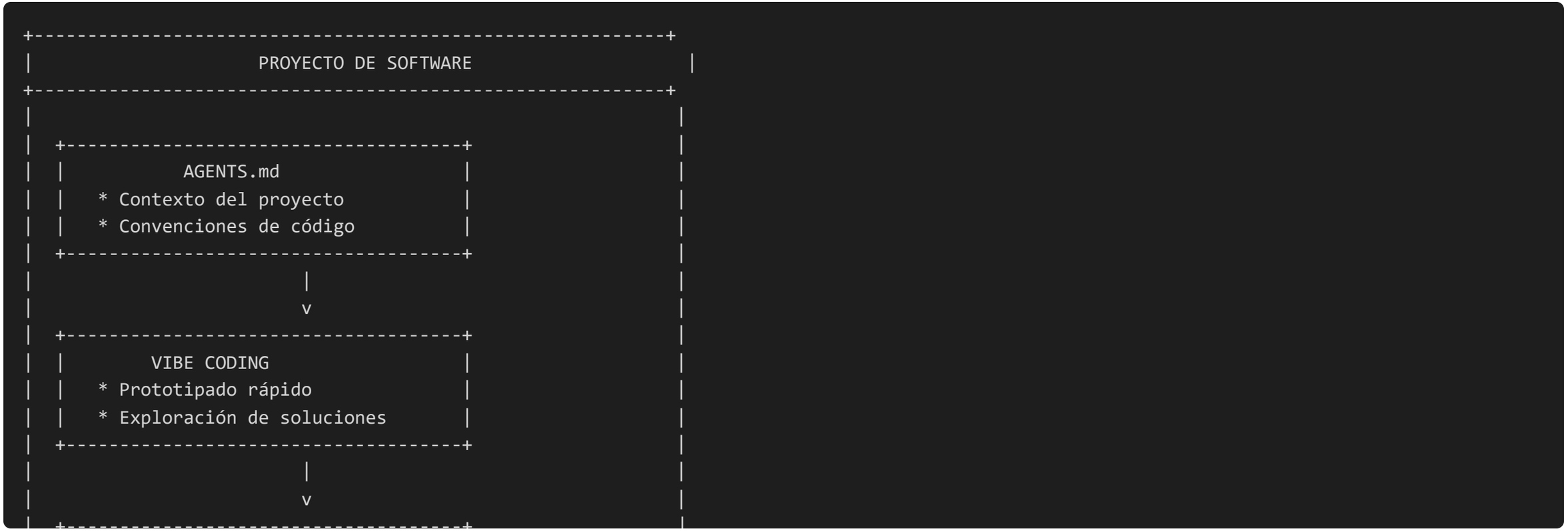
- **Agent Skills - Claude Docs**: https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview
- **Equipping agents for the real world with Agent Skills** (Anthropic Engineering): https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills
- **Claude Skills are awesome** (Simon Willison): https://simonwillison.net/2025/Oct/16/claude-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 aplicacion 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/
    └── ...                # Codigo 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	Herramienta Recomendada
Necesito validar una idea rapidamente	Vibe Coding
Voy a convertir el prototipo en producto	Spec Kit
Nuevo desarrollador o agente se une al proyecto	AGENTS.md
Necesito automatizar una tarea repetitiva	Agent Skills
Proyecto de produccion con estandares altos	Spec Kit + AGENTS.md
Equipo grande con multiples contribuidores	AGENTS.md + Agent Skills
Proyecto completo y mantenible	Los cuatro juntos

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

Ejemplo: Sistema de Notificaciones

Contexto: Desarrollo de un sistema de notificaciones para una aplicacion 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/
    └── ...                # Codigo generado
```

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	Herramienta Recomendada
Necesito validar una idea rapidamente	Vibe Coding
Voy a convertir el prototipo en producto	Spec Kit
Nuevo desarrollador o agente se une al proyecto	AGENTS.md
Necesito automatizar una tarea repetitiva	Agent Skills
Proyecto de produccion con estandares altos	Spec Kit + AGENTS.md
Equipo grande con multiples contribuidores	AGENTS.md + Agent Skills
Proyecto completo y mantenible	Los cuatro juntos

Checklist de Implementacion

Pasos practicos para comenzar

5 minutos

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

- **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

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

Comunidades en espanol:

- Reddit r/es/comments sobre programacion
- 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** - Seccion AI
- **Conferencias**: MLOps World, GenAI Summit

Comunidades en espanol:

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

Terminos Adicionales

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

Resumen y Próximos Pasos

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