netmind | a bts company

Módulo 1

Ingeniería de Prompts y Personalización para SDD

Spec-Driven Development con Copilot, Spec Kit, AGENTS.md y Agent Skills

1. Instalación y Configuración del Entorno Local

Configurar un entorno de desarrollo consistente para SDD

1.1 Introducción a la Configuración para SDD

La configuración de un entorno de desarrollo consistente es el fundamento sobre el cual se construye todo el flujo de trabajo de Spec-Driven Development.

1.1 Introducción a la Configuración para SDD

La configuración de un entorno de desarrollo consistente es el fundamento sobre el cual se construye todo el flujo de trabajo de Spec-Driven Development.

- Importancia de la configuración homogénea para colaboración y mantenimiento

1.1 Introducción a la Configuración para SDD

La configuración de un entorno de desarrollo consistente es el fundamento sobre el cual se construye todo el flujo de trabajo de Spec-Driven Development.

- Importancia de la configuración homogénea para colaboración y mantenimiento
- Los cuatro elementos del ecosistema SDD y su integración

1.1 Introducción a la Configuración para SDD

La configuración de un entorno de desarrollo consistente es el fundamento sobre el cual se construye todo el flujo de trabajo de Spec-Driven Development.

- Importancia de la configuración homogénea para colaboración y mantenimiento
- Los cuatro elementos del ecosistema SDD y su integración
- Cómo la configuración asegura coherencia en las interacciones con IA

1.1 Introducción a la Configuración para SDD

La configuración de un entorno de desarrollo consistente es el fundamento sobre el cual se construye todo el flujo de trabajo de Spec-Driven Development.

- Importancia de la configuración homogénea para colaboración y mantenimiento
- Los cuatro elementos del ecosistema SDD y su integración
- Cómo la configuración asegura coherencia en las interacciones con IA
- Relación entre configuración y productividad en equipo

1.1 Introducción a la Configuración para SDD

La configuración de un entorno de desarrollo consistente es el fundamento sobre el cual se construye todo el flujo de trabajo de Spec-Driven Development.

- Importancia de la configuración homogénea para colaboración y mantenimiento
- Los cuatro elementos del ecosistema SDD y su integración
- Cómo la configuración asegura coherencia en las interacciones con IA
- Relación entre configuración y productividad en equipo

El ecosistema SDD se compone de cuatro elementos principales:

Elemento	Propósito	Integración con SDD
GitHub Copilot	Generación de código asistida por IA	Fuente principal de sugerencias y completaciones
Spec Kit	Desarrollo dirigido por especificaciones	Flujo estructurado: specify → plan → tasks → implement
AGENTS.md	Archivo de contexto del proyecto	Proporciona instrucciones permanentes al agente
Agent Skills	Capacidades especializadas	Habilidades modulares para tareas específicas

1.2 Instalación de Copilot en Visual Studio 2022

Requisitos previos

1.2 Instalación de Copilot en Visual Studio 2022

Requisitos previos

- **Visual Studio 2022** versión 17.6 o superior (las versiones anteriores no soportan Copilot)

1.2 Instalación de Copilot en Visual Studio 2022

Requisitos previos

- **Visual Studio 2022** versión 17.6 o superior (las versiones anteriores no soportan Copilot)
- **Cuenta de GitHub** con licencia de Copilot activa (individual o empresarial)

1.2 Instalación de Copilot en Visual Studio 2022

Requisitos previos

- **Visual Studio 2022** versión 17.6 o superior (las versiones anteriores no soportan Copilot)
- **Cuenta de GitHub** con licencia de Copilot activa (individual o empresarial)
- **Conexión a internet** activa para la autenticación inicial

1.2 Instalación de Copilot en Visual Studio 2022

Requisitos previos

- **Visual Studio 2022** versión 17.6 o superior (las versiones anteriores no soportan Copilot)
- **Cuenta de GitHub** con licencia de Copilot activa (individual o empresarial)
- **Conexión a internet** activa para la autenticación inicial
- **Windows 10/11** o **Windows Server 2019+** como sistema operativo

Proceso de instalación

1. Abrir Visual Studio 2022
2. Ir a Extensions → Manage Extensions
3. En la barra de búsqueda, escribir "GitHub Copilot"
4. Seleccionar "GitHub Copilot" de los resultados
5. Hacer clic en "Download"
6. Cerrar Visual Studio para iniciar la instalación
7. El VSIX Installer se ejecutará automáticamente
8. Reiniciar Visual Studio 2022

Autenticación con GitHub

1. Ir a Tools → Options
2. Navegar a GitHub Copilot → Authentication
3. Hacer clic en "Sign in with GitHub"
4. Autorizar la aplicación en el navegador
5. Seleccionar la organización/cuenta a utilizar
6. Verificar el estado de autenticación en la barra de estado

Configuración de IntelliCode y Copilot

```
{
  "editor.inlineSuggest.enabled": true,
  "github.copilot.enable": {
    "*": true,
    "csharp": true,
    "vb": true,
    "typescript": true,
    "javascript": true,
    "typescriptreact": true,
    "javascriptreact": true
  },
  "editor.suggestOnTriggerCharacters": true,
  "editor.quickSuggestions": {
    "other": true,
    "comments": false,
    "strings": false
  }
}
```

Verificación de la instalación

Verificación de la instalación

- Crear un nuevo archivo de código C# (`.cs`)

Verificación de la instalación

- Crear un nuevo archivo de código C# (`.cs`)
- Escribir un comentario describiendo una función simple

Verificación de la instalación

- Crear un nuevo archivo de código C# (`.cs`)
- Escribir un comentario describiendo una función simple
- Esperar a que Copilot sugiera la implementación

Verificación de la instalación

- Crear un nuevo archivo de código C# (`.cs`)
- Escribir un comentario describiendo una función simple
- Esperar a que Copilot sugiera la implementación
- Aceptar con `Tab` o rechazar con `Esc`

Verificación de la instalación

- Crear un nuevo archivo de código C# (`.cs`)
- Escribir un comentario describiendo una función simple
- Esperar a que Copilot sugiera la implementación
- Aceptar con `Tab` o rechazar con `Esc`

Referencia oficial: GitHub Copilot en Visual Studio

1.3 Instalación de Copilot en Visual Studio Code

Instalación de extensiones esenciales

Extensión 1: GitHub Copilot

1. Abrir VS Code
2. Presionar Ctrl+Shift+X para abrir el panel de extensiones
3. Buscar "GitHub Copilot"
4. Seleccionar la extensión de GitHub
5. Hacer clic en "Install"
6. Reiniciar VS Code tras la instalación

Extensiones esenciales (continuación)

Extensión 2: GitHub Copilot Chat

```
1. En el panel de extensiones
2. Buscar "GitHub Copilot Chat"
3. Instalar la extensión
4. Esto habilita la ventana de chat con Copilot (Ctrl+Alt+I)
```

Extensiones esenciales (continuación)

Extensión 2: GitHub Copilot Chat

```
1. En el panel de extensiones
2. Buscar "GitHub Copilot Chat"
3. Instalar la extensión
4. Esto habilita la ventana de chat con Copilot (Ctrl+Alt+I)
```

Extensión 3: Spec Kit (opcional para CLI)

```
1. En el panel de extensiones
2. Buscar "Spec Kit" o instalar via CLI
```

Configuración por workspace y usuario

Ubicación del archivo: `.vscode/settings.json`

```
{
  "github.copilot.enable": {
    "*": true,
    "csharp": true,
    "python": true,
    "javascript": true,
    "typescript": true,
    "typescriptreact": true,
    "markdown": true
  },
  "editor.inlineSuggest.enabled": true,
  "editor.suggestOnTriggerCharacters": true,
  "editor.tabCompletion": "on",
  "files.autoSave": "afterDelay",
  "files.autoSaveDelay": 1000,
  "[markdown]": {
    "editor.wordWrap": "on",
    "editor.quickSuggestions": false
  }
```

Sincronización de configuraciones entre equipos

```
1. Presionar Ctrl+Shift+P
2. Buscar "Settings Sync: Configure"
3. Seleccionar "Settings Sync: Turn On"
4. Elegir qué sincronizar:
   - Configuración (Settings)
   - Extensiones (Extensions)
   - Fragmentos de código (Snippets)
   - Archivos de teclado (Keyboard Shortcuts)
5. Iniciar sesión con cuenta de GitHub/Microsoft
```

Preparación para AGENTS.md y Agent Skills

Habilitar lectura de AGENTS.md:

```
{
  "github.copilot.chat.attribution.enable": true,
  "github.copilot.chat.userPrompting.autoSendMemberPrivacyInformation": true
}
```

Preparación para AGENTS.md y Agent Skills

Habilitar lectura de AGENTS.md:

```
{
  "github.copilot.chat.attribution.enable": true,
  "github.copilot.chat.userPrompting.autoSendMemberPrivacyInformation": true
}
```

Verificar reconocimiento:

```
1. Crear un archivo AGENTS.md en la raíz del proyecto
2. Abrir el chat de Copilot (Ctrl+Alt+I)
3. El agente debería reconocer y usar el contexto del archivo
```

1.4 Configuraciones Avanzadas para SDD

Estructura de carpetas para Spec Kit

```
mi-proyecto-sdd/
├── .specify/                    # Configuración de Spec Kit
│   ├── memory/
│   │   └── constitution.md     # Principios y restricciones del proyecto
│   ├── scripts/
│   │   ├── check-prerequisites.sh
│   │   ├── common.sh
│   │   └── create-new-feature.sh
│   └── templates/
│       ├── CLAUDE-template.md
│       ├── plan-template.md
│       ├── spec-template.md
│       └── tasks-template.md
├── specs/                       # Especificaciones por característica
│   └── 001-nombre-caracteristica/
│       ├── plan.md
│       ├── spec.md
│       └── tasks.md
├── .github/
```

Configuración de AGENTS.md en proyectos

AGENTS.md es el archivo central que proporciona contexto al agente de IA sobre el proyecto. Debe ubicarse en la raíz del proyecto.

Configuración de AGENTS.md en proyectos

AGENTS.md es el archivo central que proporciona contexto al agente de IA sobre el proyecto. Debe ubicarse en la raíz del proyecto.

Prompt para generar un AGENTS.md básico:

```
"Genera un archivo AGENTS.md para un proyecto de API REST con las siguientes características:
- Stack: C# con ASP.NET Core 8, Entity Framework Core, SQL Server
- Patrón arquitectónico: Clean Architecture
- Estructura de carpetas: src/Domain, src/Application, src/Infrastructure, src/API
- Convenciones de código: Microsoft C# Coding Conventions
- Testing: xUnit con Moq
- Incluya secciones de Persona, Comandos, Estructura del Proyecto, Estándares de Código y Límites"
```

Configuración de AGENTS.md en proyectos

AGENTS.md es el archivo central que proporciona contexto al agente de IA sobre el proyecto. Debe ubicarse en la raíz del proyecto.

Prompt para generar un AGENTS.md básico:

```
"Genera un archivo AGENTS.md para un proyecto de API REST con las siguientes características:
- Stack: C# con ASP.NET Core 8, Entity Framework Core, SQL Server
- Patrón arquitectónico: Clean Architecture
- Estructura de carpetas: src/Domain, src/Application, src/Infrastructure, src/API
- Convenciones de código: Microsoft C# Coding Conventions
- Testing: xUnit con Moq
- Incluya secciones de Persona, Comandos, Estructura del Proyecto, Estándares de Código y Límites"
```

Prompt para generar un AGENTS.md para proyecto polyglot:

```
"Crea un archivo AGENTS.md para un proyecto con:
- Backend: Python con FastAPI, SQLAlchemy, PostgreSQL
- Frontend: TypeScript con React 18, Vite, Tailwind CSS
- Mobile: React Native con Expo
- DevOps: Docker, Azure DevOps
- Incluya Agent Skills para API generation, React components y database migrations"
```

Integración con Git y control de versiones

1. Configurar .gitignore para artefactos SDD:

```
# Spec Kit
.specify/cache/
.specify/logs/

# Agent Skills (si son personales)
~/.copilot/skills/

# Archivos temporales de prompts
*.prompt.md.bak
```

Integración con Git y control de versiones

1. Configurar .gitignore para artefactos SDD:

```
# Spec Kit
.specify/cache/
.specify/logs/

# Agent Skills (si son personales)
~/.copilot/skills/

# Archivos temporales de prompts
*.prompt.md.bak
```

2. Incluir archivos SDD en el repositorio:

```
# No ignorar estos archivos SDD
 !AGENTS.md
 !.specify/
 !specs/
 !.github/agents/
 !.github/skills/
 !CLAUDE.md
```

Git hook para validación (opcional)

```
# Crear pre-commit hook para validar estructura SDD
cat > .git/hooks/pre-commit << 'EOF'
#!/bin/bash
# Validar que specs/ tiene archivos spec.md, plan.md, tasks.md
for dir in specs/*/; do
  if [ ! -f "$dir/spec.md" ]; then
    echo "Error: Falta spec.md en $dir"
    exit 1
  fi
done
echo "Validación SDD completada"
EOF
chmod +x .git/hooks/pre-commit
```

Exclusión de archivos sensibles y specs internas

1. Archivos sensibles a excluir:

```
// .vscode/settings.json
{
  "files.exclude": {
    "**/*.key": true,
    "**/*.pem": true,
    "**/.env*": true,
    "**/secrets.json": true,
    "**/connectionStrings.secret.json": true,
    "**/credentials.json": true
  }
}
```

Exclusión de archivos sensibles y specs internas

1. Archivos sensibles a excluir:

```
// .vscode/settings.json
{
  "files.exclude": {
    "**/*.key": true,
    "**/*.pem": true,
    "**/.env*": true,
    "**/secrets.json": true,
    "**/connectionStrings.secret.json": true,
    "**/credentials.json": true
  }
}
```

2. Specs internas que no deben procesarse:

```
# Marcar specs como no procesables
specs/000-borrador-pendiente-review/
specs/999-experimentales/
```

Exclusión de archivos sensibles y specs internas

1. Archivos sensibles a excluir:

```
// .vscode/settings.json
{
  "files.exclude": {
    "**/*.key": true,
    "**/*.pem": true,
    "**/.env*": true,
    "**/secrets.json": true,
    "**/connectionStrings.secret.json": true,
    "**/credentials.json": true
  }
}
```

2. Specs internas que no deben procesarse:

```
# Marcar specs como no procesables
specs/000-borrador-pendiente-review/
specs/999-experimentales/
```

3. Configuración de Agent Skills personales:

```
# Ubicación de Agent Skills personales
Windows: %APPDATA%/Copilot/skills/
Linux/macOS: ~/.copilot/skills/
```

2. Fundamentos de Prompting Efectivo para SDD

Dominar técnicas de prompting estructurado para Spec-Driven Development

2.1 Principios de Prompting Efectivo en SDD

El prompting efectivo en Spec-Driven Development difiere significativamente del prompting tradicional utilizado en Vibe Coding.

2.1 Principios de Prompting Efectivo en SDD

El prompting efectivo en Spec-Driven Development difiere significativamente del prompting tradicional utilizado en Vibe Coding.

- Diferencia entre Vibe Coding (fluido/exploratorio) y SDD (estructurado/específico)

2.1 Principios de Prompting Efectivo en SDD

El prompting efectivo en Spec-Driven Development difiere significativamente del prompting tradicional utilizado en Vibe Coding.

- Diferencia entre Vibe Coding (fluido/exploratorio) y SDD (estructurado/específico)
- Equilibrio entre especificidad y contextualidad

2.1 Principios de Prompting Efectivo en SDD

El prompting efectivo en Spec-Driven Development difiere significativamente del prompting tradicional utilizado en Vibe Coding.

- Diferencia entre Vibe Coding (fluido/exploratorio) y SDD (estructurado/específico)
- Equilibrio entre especificidad y contextualidad
- Ciclo de mejora continua con iteración y refinamiento

2.1 Principios de Prompting Efectivo en SDD

El prompting efectivo en Spec-Driven Development difiere significativamente del prompting tradicional utilizado en Vibe Coding.

- Diferencia entre Vibe Coding (fluido/exploratorio) y SDD (estructurado/específico)
- Equilibrio entre especificidad y contextualidad
- Ciclo de mejora continua con iteración y refinamiento
- Importancia de la trazabilidad en prompts

Especificidad versus contextualidad en SDD

Ejemplo de prompt no específico (Vibe Coding tradicional):

```
"Crea una función para validar emails"
```

Especificidad versus contextualidad en SDD

Ejemplo de prompt no específico (Vibe Coding tradicional):

```
"Crea una función para validar emails"
```

Ejemplo de prompt específico para SDD:

```
"Crea una función de validación de emails para el módulo de autenticación de PortalEmpleo API con las siguientes especificaciones:

## Requisitos Funcionales
- Validar formato RFC 5322 estándar de emails
- Longitud máxima: 254 caracteres
- Permitir caracteres especiales en dominio: -, _
- Rechazar emails con doble @ consecutivos
- Rechazar emails que empiezan o terminan con punto

## Contexto del Proyecto
- Lenguaje: C# 12
- Framework: ASP.NET Core 8
- Patrón: Repository con Dependency Injection
- Ubicación: src/Domain/ValueObjects/Email.cs
- Convenciones: Microsoft C# Coding Conventions

## Criterios de Aceptación
- Tests unitarios cubriendo casos válidos e inválidos
- Mensajes de error específicos para cada tipo de validación fallida
```

Iteración y refinamiento de prompts contra specs

```
graph TD
    A[Definir especificación] --> B[Crear prompt inicial]
    B --> C[Generar código]
    C --> D[Validar contra spec]
    D --> E{¿Cumple criterios?}
    E -->|No| F[Identificar gaps]
    F --> G[Refinar prompt]
    G --> B
    E -->|Sí| H[Documentar decisión]
```

Iteración y refinamiento de prompts contra specs

```
graph TD
    A[Definir especificación] --> B[Crear prompt inicial]
    B --> C[Generar código]
    C --> D[Validar contra spec]
    D --> E{¿Cumple criterios?}
    E -->|No| F[Identificar gaps]
    F --> G[Refinar prompt]
    G --> B
    E -->|Sí| H[Documentar decisión]
```

Plantilla de prompt iterativo:

```
"Basándote en la especificación en specs/001-auth-api/spec.md y el feedback anterior:

## Feedback Anterior
[Descripción de qué no funcionó]

## Especificación de Referencia
- Requisito: [ID del requisito]
- Criterios de aceptación: [lista]

## Refinamiento Solicitado
[Cambios específicos o adiciones]

## Validación Requerida
[Cómo verificar que el resultado es correcto]"
```

Control de output alineado con agentes definidos

Prompt para docs-agent:

```
"@docs-agent Genera documentación API para el endpoint POST /api/users
siguiendo las convenciones definidas en AGENTS.md.
Incluye:
- Descripción del endpoint
- Parámetros de request
- Schema de request body
- Códigos de respuesta
- Ejemplo de curl
- Notas de autenticación requerida"
```

Control de output alineado con agentes definidos

Prompt para docs-agent:

```
"@docs-agent Genera documentación API para el endpoint POST /api/users
siguiendo las convenciones definidas en AGENTS.md.
Incluye:
- Descripción del endpoint
- Parámetros de request
- Schema de request body
- Códigos de respuesta
- Ejemplo de curl
- Notas de autenticación requerida"
```

Prompt para test-agent:

```
"@test-agent Escribe tests unitarios para la clase EmailValidator
usando xUnit y Moq, siguiendo el template en .specify/templates/test-template.md.
Cobertura requerida:
- Emails válidos
- Emails inválidos por formato
- Emails que exceden longitud
- Casos edge (null, empty, whitespace)"
```

Formato de respuesta estructurado para validación

Prompt con formato estructurado:

````
"Genera la implementación siguiendo este formato de respuesta:

## Código Implementado
```csharp
[código completo]
```

## Trazabilidad
| Requisito SDD | Implementado |
|---------------|--------------|
| REQ-AUTH-001 | [línea/método] |
| REQ-AUTH-002 | [línea/método] |

## Validación
- [ ] Cumple criterio de aceptación 1
- [ ] Cumple criterio de aceptación 2

## Archivos Modificados
- src/Domain/ValueObjects/Email.cs
````

2.2 Patrones de Prompting para SDD

Esta subsección presenta patrones estructurados para las tareas más comunes en SDD: generación, validación, refactoring y documentación.

2.2 Patrones de Prompting para SDD

Esta subsección presenta patrones estructurados para las tareas más comunes en SDD: generación, validación, refactoring y documentación.

- Transformación de especificaciones formales en código

2.2 Patrones de Prompting para SDD

Esta subsección presenta patrones estructurados para las tareas más comunes en SDD: generación, validación, refactoring y documentación.

- Transformación de especificaciones formales en código
- Verificación de cumplimiento de specs

2.2 Patrones de Prompting para SDD

Esta subsección presenta patrones estructurados para las tareas más comunes en SDD: generación, validación, refactoring y documentación.

- Transformación de especificaciones formales en código
- Verificación de cumplimiento de specs
- Refactoring sin alterar comportamiento observable

2.2 Patrones de Prompting para SDD

Esta subsección presenta patrones estructurados para las tareas más comunes en SDD: generación, validación, refactoring y documentación.

- Transformación de especificaciones formales en código
- Verificación de cumplimiento de specs
- Refactoring sin alterar comportamiento observable
- Documentación técnica desde specs

Prompts de generación desde especificaciones

Estructura del patrón:

```
1. Referenciar la especificación fuente
2. Proporcionar contexto técnico del proyecto
3. Especificar restricciones arquitectónicas
4. Definir formato de salida
5. Solicitar trazabilidad a requisitos
```

Prompts de generación desde especificaciones

Estructura del patrón:

```
1. Referenciar la especificación fuente
2. Proporcionar contexto técnico del proyecto
3. Especificar restricciones arquitectónicas
4. Definir formato de salida
5. Solicitar trazabilidad a requisitos
```

Prompt de generación desde spec.md:

```
"Genera la implementación completa del endpoint GET /api/tareas/{id}
según la especificación en specs/001-task-api/spec.md#endpoint-get-tarea

## Especificación de Referencia
- Sección: 2.1.3 GET /api/tareas/{id}
- Criterios de aceptación: CA-TASK-003, CA-TASK-004

## Contexto Técnico
- Controller: src/API/Controllers/TareasController.cs
- Service: src/Application/Services/TareaService.cs
- Repository: src/Infrastructure/Data/Repositories/TareaRepository.cs
- Response DTO: src/API/DTOs/TareaResponseDto.cs

## Requisitos de Implementación
1. Obtener tarea por ID con Include de UsuarioAsignado
2. Retornar 404 si no existe
3. Retornar 401 si no tiene permisos
4. Cache headers para GET
```

Prompts de validación contra specs definidas

Prompt de validación:

```
"Valida el código en src/Application/Services/TareaService.cs
contra la especificación en specs/001-task-api/spec.md

## Checklist de Validación
- [ ] Método CreateTarea implementa validación de negocio
- [ ] UpdateTarea verifica ownership
- [ ] DeleteTarea implementa soft delete
- [ ] GetAllTareas aplica filtros de usuario
- [ ] Maneja excepciones según Error Handling Policy

## Criterios de Aceptación a Verificar
CA-TASK-001: Tarea creada con ID único en < 50ms
CA-TASK-002: No permite crear tareas duplicadas por título
CA-TASK-005: Lista filtrada visible solo para usuarios con acceso

## Reporte de Validación
```
Cumplimiento
- Total requisitos: X
```
```

Prompts de refactoring manteniendo especificaciones

Prompt de refactoring:

```
"Realiza refactoring de src/Domain/Entities/Tarea.cs
para aplicar principios SOLID mientras mantienes
compatibilidad con specs/001-task-api/spec.md

## Reglas de Refactoring
1. Single Responsibility: Extraer value objects
2. Open/Closed: Preparar para extensión
3. Dependency Inversion: Usar interfaces
4. No cambiar comportamiento público

## Validación Post-Refactoring
- Tests existentes deben pasar
- API contract unchanged
- Performance no degradado > 5%

## Salida Esperada
```
Cambios Realizados
- Archivo: Tarea.cs
```
```

Prompts de documentación técnica SDD

Prompt para documentar especificación:

```
"@docs-agent Genera documentación técnica para specs/001-task-api/
siguiendo el template en .specify/templates/docs-template.md

## Documentos a Generar
1. README.md - Visión general del feature
2. API.md - Documentación de endpoints
3. DATA_MODEL.md - Esquema de datos

## Incluir
- Diagrama de secuencia para operaciones principales
- Tabla de decisiones de diseño (ADR)
- Matriz de trazabilidad req→test
- Notas de implementación para desarrolladores"
```

Prompts de documentación técnica SDD

Prompt para documentar especificación:

```
"@docs-agent Genera documentación técnica para specs/001-task-api/
siguiendo el template en .specify/templates/docs-template.md

## Documentos a Generar
1. README.md - Visión general del feature
2. API.md - Documentación de endpoints
3. DATA_MODEL.md - Esquema de datos

## Incluir
- Diagrama de secuencia para operaciones principales
- Tabla de decisiones de diseño (ADR)
- Matriz de trazabilidad req→test
- Notas de implementación para desarrolladores"
```

Prompt que referencia AGENTS.md:

```
"@api-agent Implementa el endpoint POST /api/tareas
siguiendo las convenciones del proyecto definidas en AGENTS.md:
- Estándares de código: Microsoft C# Conventions
- Patrón: Controller → Service → Repository
- Error handling: ProblemDetails format
- Testing: xUnit con InMemory database

Especificación técnica: specs/001-task-api/spec.md
Sección: 2.1.1 POST /api/tareas"
```

2.3 Técnicas Avanzadas de Control

Esta subsección profundiza en técnicas avanzadas para obtener código que cumpla estándares, patrones arquitectónicos y convenciones específicas del proyecto.

Especificación de estándares de código en prompts

Prompt con estándares detallados:

````
"Implementa el nuevo endpoint siguiendo estos estándares:

## Estándares de Código C#
- Nomenclatura: PascalCase para clases/métodos, camelCase para variables
- Visibilidad: private explícito para campos
- Async: suffix 'Async' para métodos asíncronos
- Nullable: habilitar annotations warnings
- Documentación: XML comments para APIs públicas

## Patrones Requeridos
```csharp
// Controller pattern
[ApiController]
[Route("api/[controller]")]
public class TareasController : ControllerBase
{
    private readonly ITareaService _service;
    public TareasController(ITareaService service) => _service = service;
}
````
````

Definición de patrones arquitectónicos SDD

Prompt con constraints arquitectónicos:

```
"Implementa con los siguientes constraints arquitectónicos:

## Clean Architecture Layers
src/
├── Domain/           # Entidades, Value Objects, Enums, Interfaces de repositorio
├── Application/      # Services, DTOs, Interfaces de servicio, Behaviors
├── Infrastructure/   # Repositories, External services, EF Core
└── API/              # Controllers, Filters, Middleware

## Reglas de Dependencia
- Domain no depende de nadie
- Application depende solo de Domain
- Infrastructure depende de Application y Domain
- API depende de Application

## Cross-Cutting Concerns
- Logging: ILogger<t>
- Validation: FluentValidation
- Error Handling: ExceptionHandler middleware
```

Establecimiento de convenciones de nomenclatura

Prompt con convenciones:

```
"Usa estas convenciones de nomenclatura:

## Clases y Tipos
- Entidades: TareaEntity, UsuarioEntity
- DTOs: TareaCreateDto, TareaResponseDto
- Services: TareaService, ITareaService
- Repositories: TareaRepository, ITareaRepository
- Validators: CreateTareaValidator, UpdateTareaValidator

## Métodos
- CRUD: GetAll(), GetById(id), Create(dto), Update(id, dto), Delete(id)
- Queries: GetTareasPendientes(), GetTareasPorUsuario(usuarioId)
- Commands: CreateTareaCommand, CompleteTareaCommand

## Variables y Parámetros
- Singular para entidades: Tarea tarea
- Plural para colecciones: List<tarea> tareas
- Prefijo para DTOs: createTareaDto, updateTareaDto
```

Uso de Agent Skills en prompts especializados

Prompt invocando Agent Skill:

```
"@agent Usa la skill webapp-testing para crear tests E2E
del flujo de creación de tareas en PortalEmpleo.

## Agent Skill a Usar
- Skill: .github/skills/e2e-testing/
- Template: test-template.spec.ts
- Navegador: Playwright con Chromium

## Escenarios a Testear
1. Usuario logueado puede crear tarea
2. Usuario no logueado recibe 401
3. Datos inválidos muestran errores de validación
4. Tarea aparece en lista inmediatamente

## Datos de Test
- Usuario test: testuser@portalempleo.com
- Permisos: employee"
```

Patrones y técnicas SDD - Resumen

Patrón	Propósito	Cuándo Usar
Generación desde specs	spec → código	Nuevas features
Validación contra specs	Verificar cumplimiento	Antes de commit
Refactoring	Mejorar sin alterar	Mantenimiento
Documentación	Generar docs	Nuevas APIs
Agent Skills	Procedimientos especializados	Tareas específicas

3. Personalización Avanzada y Mejores Prácticas con SDD

Configurar Spec Kit, AGENTS.md y Agent Skills según necesidades del proyecto

3.1 Instalación de Spec Kit, AGENTS.md y Agent Skills

Estructura y componentes de Spec Kit

Spec Kit es un toolkit de código abierto desarrollado por GitHub que implementa la metodología de Spec-Driven Development.

3.1 Instalación de Spec Kit, AGENTS.md y Agent Skills

Estructura y componentes de Spec Kit

Spec Kit es un toolkit de código abierto desarrollado por GitHub que implementa la metodología de Spec-Driven Development.

Instalación de Spec Kit:

```
# Método 1: Instalación persistente con uv (recomendado)
uv tool install specify-cli --from git+https://github.com/github/spec-kit.git

# Verificar instalación
specify check

# Actualizar Spec Kit
uv tool install specify-cli --force --from git+https://github.com/github/spec-kit.git

# Método 2: Uso único con uvx
uvx --from git+https://github.com/github/spec-kit.git specify init <project_name></project_name>
```

Inicialización de proyecto con Spec Kit

```
# Inicializar nuevo proyecto
specify init mi-proyecto-sdd

# Especificar agente de IA a usar
specify init mi-proyecto-sdd --ai copilot
specify init mi-proyecto-sdd --ai claude

# Inicializar en directorio actual
specify init . --ai copilot
specify init --here --ai copilot

# Forzar inicialización en directorio no vacío
specify init . --force --ai copilot

# Omitir inicialización de Git
specify init mi-proyecto --no-git --ai copilot

# Habilitar salida de depuración
specify init mi-proyecto --debug --ai copilot
```

Comandos principales de Spec Kit

Comando	Descripción
`/speckit.constitution`	Crea o actualiza la Constitución del proyecto
`/speckit.specify`	Genera especificación detallada a partir de descripción de alto nivel
`/speckit.clarify`	Analiza especificación y formula preguntas para identificar brechas
`/speckit.plan`	Genera plan de implementación técnica
`/speckit.tasks`	Descompone el plan en tareas discretas
`/speckit.implement`	Guía la implementación de tareas
`/speckit.analyze`	Verifica coherencia entre especificación, plan, tareas y Constitución
`/speckit.checklist`	Genera listas de comprobación de validación

Workflow de Trabajo Spec Kit

```
flowchart TD
    A[Inicio: Requisito de negocio] --> B[/specKit.specify/]
    B --> C[specs/[feature]/spec.md]
    C --> D[/specKit.plan/]
    D --> E[specs/[feature]/plan.md]
    E --> F[/specKit.tasks/]
    F --> G[specs/[feature]/tasks.md]
    G --> H[Implementación]
    H --> I[/specKit.analyze/]
    I --> J[Validación contra spec]
```

Workflow de Trabajo Spec Kit

```
flowchart TD
    A[Inicio: Requisito de negocio] --> B[/specKit.specify/]
    B --> C[specs/[feature]/spec.md]
    C --> D[/specKit.plan/]
    D --> E[specs/[feature]/plan.md]
    E --> F[/specKit.tasks/]
    F --> G[specs/[feature]/tasks.md]
    G --> H[Implementación]
    H --> I[/specKit.analyze/]
    I --> J[Validación contra spec]
```

Artefactos SDD y su Propósito:

Artefacto	Propósito	Comando Spec Kit	Siguiente paso
`spec.md`	Definición funcional del feature	`/specKit.specify`	Plan técnico
`plan.md`	Plan de implementación técnica	`/specKit.plan`	Descomposición
`tasks.md`	Tareas ejecutables discretas	`/specKit.tasks`	Implementación
`constitution.md`	Principios y estándares del proyecto	`/speckit.constitution`	Referencia continua

Configuración de AGENTS.md para diferentes roles

Dónde se guarda: `AGENTS.md` (raíz del proyecto) y `.github/agents/` para agentes especializados

Cómo se usa: Se carga automáticamente cuando Copilot analiza el proyecto. Se invoca con prefijos como `@backend-dev`, `@test-dev`, `@docs-dev` en prompts.

Configuración de AGENTS.md para diferentes roles

Dónde se guarda: `AGENTS.md` (raíz del proyecto) y `.github/agents/` para agentes especializados

Cómo se usa: Se carga automáticamente cuando Copilot analiza el proyecto. Se invoca con prefijos como `@backend-dev`, `@test-dev`, `@docs-dev` en prompts.

Prompt para generar AGENTS.md con múltiples agentes:

```
"Genera un archivo AGENTS.md para un proyecto ASP.NET Core 8 con:
- Agente @backend-dev para desarrollo de APIs C#
- Agente @frontend-dev para React TypeScript
- Agente @test-dev para generación de tests
- Agente @docs-dev para documentación técnica

Incluye para cada agente:
- Persona/rol específico
- Stack tecnológico
- Comandos disponibles
- Estándares de código
- Límites (Always/Ask/Never)"
```

Template de AGENTS.md completo (Parte 1)

```
---
name: project_agents
description: Agentes especializados para proyecto PortalEmpleo SDD
---

# PortalEmpleo - AGENTS.md

## Persona Global

You are an expert developer working on PortalEmpleo, a task management system built with Spec-Driven Development methodology. You always refer t

## Project Knowledge

### Tech Stack
- **Backend:** C# 12, ASP.NET Core 8, Entity Framework Core 8, SQL Server 2022
- **Frontend:** TypeScript 5, React 18, Vite 5, Tailwind CSS 3
- **Testing:** xUnit, Moq, Playwright
- **DevOps:** Docker, Azure DevOps, GitHub Actions
```

Template de AGENTS.md completo (Parte 2)

````
## @backend-dev

You are a backend architecture expert specializing in C# and ASP.NET Core.

### Role
- Design and implement REST APIs
- Write clean, testable C# code
- Apply Clean Architecture principles

### Commands
```bash
dotnet build          # Build solution
dotnet test           # Run unit tests
dotnet ef migrations add [Name]  # Create migration
dotnet ef database update        # Apply migrations
```

### Standards
- XML docs for public APIs
````

Setup de Agent Skills según necesidades del proyecto

Dónde se guarda: `.github/skills/[skill-name]/SKILL.md` y subdirectorios de templates

Cómo se usa: Se invocan automáticamente según el contexto del código o manualmente con `@use [skill-name]` en prompts.

Setup de Agent Skills según necesidades del proyecto

Dónde se guarda: `.github/skills/[skill-name]/SKILL.md` y subdirectorios de templates

Cómo se usa: Se invocan automáticamente según el contexto del código o manualmente con `@use [skill-name]` en prompts.

Estructura de Agent Skills:

```
.github/skills/
├── api-development/
│   ├── SKILL.md
│   └── templates/
│       ├── controller-template.cs
│       ├── dto-template.cs
│       └── validator-template.cs
├── testing/
│   ├── SKILL.md
│   └── templates/
│       ├── unit-test-template.cs
│       └── integration-test-template.cs
├── database/
│   ├── SKILL.md
│   └── scripts/
│       └── migration-template.sql
└── documentation/
    ├── SKILL.md
    └── templates/
```

Prompt para generar Agent Skill de desarrollo de APIs

```
"Crea una Agent Skill para desarrollo de APIs ASP.NET Core en
.github/skills/api-development/SKILL.md

## Contenido Requerido
1. YAML frontmatter con name y description
2. Cuándo usar esta skill
3. Procedimientos paso a paso para crear endpoint
4. Templates de código referenciados
5. Mejores prácticas de la industria

## Templates a Incluir
- Controller con atributos de documentación
- DTO con DataAnnotations
- FluentValidator
- Integration test

## Ejemplos
- Endpoint CRUD completo
- Manejo de errores con ProblemDetails
```

3.2 Personalización por Tipo de Proyecto

Proyectos web en C#: ASP.NET Core con SDD

Prompt para generar estructura de proyecto ASP.NET Core SDD:

```
"Genera la estructura de proyecto ASP.NET Core 8 con SDD:

## Requisitos
- Clean Architecture con spec-kit
- AGENTS.md con agente @backend-dev
- Agent Skills para API development y testing
- Templates para controllers, DTOs, validators
- Integración con SQL Server

## Estructura a Generar
src/
├── Domain/
│   ├── Entities/
│   ├── ValueObjects/
│   ├── Enums/
│   └── Interfaces/
├── Application/
│   ├── DTOs/
│   ├── Interfaces/
```

Proyectos de data: Python notebooks y scripts SDD

Prompt para generar estructura de proyecto Python SDD:

```
"Genera estructura de proyecto Python para data science con SDD:

## Stack
- Python 3.12
- Jupyter Notebooks
- Pandas, NumPy, Scikit-learn
- MLflow para tracking

## Estructura
src/
├── data/
│   ├── ingestion/
│   ├── processing/
│   └── features/
├── models/
│   ├── training/
│   └── evaluation/
└── utils/
notebooks/
```

Proyectos mobile: React Native componentes SDD

Prompt para generar estructura React Native SDD:

```
"Genera estructura de proyecto React Native con SDD:

## Stack
- React Native con Expo
- TypeScript 5
- React Query para data fetching
- React Navigation

## Estructura
src/
├── components/
│   ├── common/
│   ├── forms/
│   └── layouts/
├── screens/
│   ├── auth/
│   ├── home/
│   └── settings/
├── hooks/
```

Adaptación de AGENTS.md por tecnología

Prompt para generar AGENTS.md específico por tecnología:

```
"Genera una versión específica de AGENTS.md para:

1. Proyecto C# ASP.NET Core:
   - Incluir comandos .NET CLI
   - Estándares Microsoft C# Conventions
   - Entity Framework patterns

2. Proyecto Python:
   - Incluir comandos pip/uv
   - PEP 8 standards
   - Virtual environment setup

3. Proyecto React Native:
   - Incluir comandos npm/expo
   - TypeScript strict mode
   - React hooks best practices"
```

3.3 Establecimiento de Convenciones SDD

Dónde se guarda: `.specify/templates/`, `.specify/memory/`, `.github/workflows/`

Cómo se usa: Estos artefactos se referencian automáticamente en prompts y tasks.

Templates de prompts compartidos para SDD

Artifacto: `.specify/templates/feature-prompt-template.md`

Templates de prompts compartidos para SDD

Artifacto: `.specify/templates/feature-prompt-template.md`

Template de prompt para nueva feature:

```
## [Feature Name]

### Especificación de Referencia
- Archivo: specs/[feature-id]/spec.md
- Requisitos: [lista de requisitos]

### Contexto del Proyecto
- Stack: [tecnologías]
- Patrón: [arquitectura]
- Ubicación: [ruta src/]

### Criterios de Aceptación
- [ ] Criterio 1
- [ ] Criterio 2

### Restricciones
- [restricción 1]
- [restricción 2]
```

Patrones de refactoring y workflows documentados

Patrones de refactoring documentados:

```
## Refactoring Patterns

### Extract Method
**Cuándo usar:** Método > 20 líneas o múltiples responsabilidades
**Proceso:**
1. Identificar bloque lógico
2. Crear método con nombre descriptivo
3. Mover bloque al nuevo método
4. Reemplazar con llamada
5. Verificar tests

### Replace Conditional with Polymorphism
**Cuándo usar:** Switch/if-else sobre tipo en múltiples lugares
**Proceso:**
1. Identificar jerarquía de tipos
2. Crear interfaz/base class
3. Mover comportamiento a implementaciones
4. Reemplazar condicionales con polimorfismo
```

Workflow SDD con Scrum

```
## Sprint SDD Workflow

### Sprint Planning (Día 1)
1. [ ] Review backlog de user stories
2. [ ] Seleccionar stories para el sprint
3. [ ] Crear specs/ para cada story
4. [ ] Generar plans y tasks

### Daily (Días 2-5)
1. [ ] Tomar task de tasks.md
2. [ ] Implementar según spec
3. [ ] Validar contra criterios de aceptación
4. [ ] Actualizar estado en tasks.md

### Sprint Review (Día 6)
1. [ ] Demo de features completados
2. [ ] Validación contra specs originales
3. [ ] Documentar aprendizajes
```

4. Ejercicio de Consolidación: Migración a SDD

Aplicación al Caso Práctico PortalEmpleo

4.1 Aplicación al Caso Práctico PortalEmpleo

Objetivo: Configurar el entorno SDD completo para el proyecto PortalEmpleo API.

4.1 Aplicación al Caso Práctico PortalEmpleo

Objetivo: Configurar el entorno SDD completo para el proyecto PortalEmpleo API.

Entregables del Módulo 1 (Caso Práctico):

Artefacto	Ubicación	Descripción
`AGENTS.md`	Raíz del proyecto	Agentes especializados: @backend-dev, @data-dev, @test-dev
`.specify/memory/constitution.md`	`.specify/memory/`	Principios y estándares del proyecto
`.specify/templates/prompts/`	`.specify/templates/`	Biblioteca de 15+ prompts reutilizables
`.vscode/settings.json`	`.vscode/`	Configuración Copilot optimizada

4.1 Aplicación al Caso Práctico PortalEmpleo

Objetivo: Configurar el entorno SDD completo para el proyecto PortalEmpleo API.

Entregables del Módulo 1 (Caso Práctico):

Artefacto	Ubicación	Descripción
`AGENTS.md`	Raíz del proyecto	Agentes especializados: @backend-dev, @data-dev, @test-dev
`.specify/memory/constitution.md`	`.specify/memory/`	Principios y estándares del proyecto
`.specify/templates/prompts/`	`.specify/templates/`	Biblioteca de 15+ prompts reutilizables
`.vscode/settings.json`	`.vscode/`	Configuración Copilot optimizada

Conexión con el caso práctico:

El caso práctico PortalEmpleo define los requisitos funcionales (16 endpoints) y no funcionales (Clean Architecture, JWT, etc.) que usaremos a lo largo del curso.

Checklist de implementación SDD en proyecto brownfield

```
## Checklist: Migración a SDD

### Fase 1: Preparación
- [ ] Evaluar complejidad del proyecto
- [ ] Identificar áreas prioritarias para SDD
- [ ] Obtener buy-in del equipo

### Fase 2: Configuración Base
- [ ] Instalar Spec Kit
- [ ] Crear AGENTS.md inicial
- [ ] Configurar estructura specs/
- [ ] Crear constitution.md con principios

### Fase 3: Documentación
- [ ] Documentar arquitectura actual
- [ ] Identificar patrones de código existentes
- [ ] Crear templates de refactoring
- [ ] Documentar convenciones en AGENTS.md
```

Caso de estudio: PortalEmpleo

El proyecto PortalEmpleo es un proyecto greenfield (nuevo), por lo que aplicaremos la configuración SDD desde el inicio.

Caso de estudio: PortalEmpleo

El proyecto PortalEmpleo es un proyecto greenfield (nuevo), por lo que aplicaremos la configuración SDD desde el inicio.

> **Nota:** Las fases de migración son relevantes para equipos que adopten SDD en proyectos existentes (brownfield).

Caso de estudio: PortalEmpleo

El proyecto PortalEmpleo es un proyecto greenfield (nuevo), por lo que aplicaremos la configuración SDD desde el inicio.

> **Nota:** Las fases de migración son relevantes para equipos que adopten SDD en proyectos existentes (brownfield).

Véase también:

- Sección 5.1 del caso práctico para entregables específicos del Módulo 1
- Sección 1.3 para instalación de Spec Kit, AGENTS.md y Agent Skills

5. Referencias Oficiales

Recursos oficiales del Módulo 1

Referencias del Módulo 1

GitHub Copilot

- Documentación oficial
- GitHub Copilot en VS Code
- GitHub Copilot en Visual Studio

Referencias del Módulo 1

GitHub Copilot

- Documentación oficial
- GitHub Copilot en VS Code
- GitHub Copilot en Visual Studio

Spec Kit (Spec-Driven Development)

- Repositorio oficial
- Microsoft Learn - Spec-Driven Development
- GitHub Blog - Spec-Driven Development
- Microsoft DevBlogs - Spec Kit

Referencias (continuación)

AGENTS.md

- GitHub Blog - How to write a great agents.md
- Sitio oficial AGENTS.md
- Repositorio principal

Referencias (continuación)

AGENTS.md

- GitHub Blog - How to write a great agents.md
- Sitio oficial AGENTS.md
- Repositorio principal

Agent Skills

- Documentación Claude - Agent Skills
- Anthropic Engineering - Agent Skills

Referencias (continuación)

AGENTS.md

- GitHub Blog - How to write a great agents.md
- Sitio oficial AGENTS.md
- Repositorio principal

Agent Skills

- Documentación Claude - Agent Skills
- Anthropic Engineering - Agent Skills

Prompt Engineering

- Prompt Engineering Guide
- OpenAI Best Practices

Resumen del Módulo 1

Resumen del Módulo 1

Sección	Conceptos Clave	Herramientas/Artefactos
1.1 Instalación	Configuración consistente del entorno	VS Code, VS 2022, Copilot, Spec Kit, estructura de carpetas
1.2 Prompting	Ingeniería de prompts para SDD	Prompts estructurados, validación contra specs, patrones de prompting
1.3 Personalización	Adaptación de herramientas al proyecto	AGENTS.md, Agent Skills, templates, convenciones por tecnología
1.4 Consolidación	Configuración PortalEmpleo para SDD	AGENTS.md, constitution.md, templates, checklist brownfield

Resumen del Módulo 1

Sección	Conceptos Clave	Herramientas/Artefactos
1.1 Instalación	Configuración consistente del entorno	VS Code, VS 2022, Copilot, Spec Kit, estructura de carpetas
1.2 Prompting	Ingeniería de prompts para SDD	Prompts estructurados, validación contra specs, patrones de prompting
1.3 Personalización	Adaptación de herramientas al proyecto	AGENTS.md, Agent Skills, templates, convenciones por tecnología
1.4 Consolidación	Configuración PortalEmpleo para SDD	AGENTS.md, constitution.md, templates, checklist brownfield

Conceptos clave del módulo:

- Ecosistema SDD de 4 elementos: Copilot, Spec Kit, AGENTS.md, Agent Skills
- Diferencia entre Vibe Coding (fluido) y SDD (estructurado)
- Patrones de prompting: generación, validación, refactoring, documentación
- Workflow Spec Kit: specify → plan → tasks → implement → analyze
- Configuración por tipo de proyecto (C#, Python, React Native)

Resumen del Módulo 1

Sección	Conceptos Clave	Herramientas/Artefactos
1.1 Instalación	Configuración consistente del entorno	VS Code, VS 2022, Copilot, Spec Kit, estructura de carpetas
1.2 Prompting	Ingeniería de prompts para SDD	Prompts estructurados, validación contra specs, patrones de prompting
1.3 Personalización	Adaptación de herramientas al proyecto	AGENTS.md, Agent Skills, templates, convenciones por tecnología
1.4 Consolidación	Configuración PortalEmpleo para SDD	AGENTS.md, constitution.md, templates, checklist brownfield

Conceptos clave del módulo:

- Ecosistema SDD de 4 elementos: Copilot, Spec Kit, AGENTS.md, Agent Skills
- Diferencia entre Vibe Coding (fluido) y SDD (estructurado)
- Patrones de prompting: generación, validación, refactoring, documentación
- Workflow Spec Kit: specify → plan → tasks → implement → analyze
- Configuración por tipo de proyecto (C#, Python, React Native)

Próximo paso: El Módulo 2 integra estas herramientas en el ciclo de vida SDD completo.

Gracias

Módulo 1 Completado

Siguiente: Modulo 2 - Microsoft Copilot en el Ciclo de Vida