netmind | a bts company

# Módulo 2

## Microsoft Copilot en el Ciclo de Vida SDD

Integración de Copilot y Spec Kit en las fases de diseño y desarrollo

# Prerrequisitos y Contexto

Workflow Spec Kit para el Módulo 2

# Artefactos SDD Disponibles

| Artefacto | Ubicación | Propósito |
|---|---|---|
| `constitution.md` | `.specify/memory/` | Principios, estándares y convenciones del proyecto |
| `spec.md` | `specs/[feature]/` | Especificación funcional con requisitos y criterios de aceptación |
| `plan.md` | `specs/[feature]/` | Plan técnico con decisiones de arquitectura |
| `tasks.md` | `specs/[feature]/` | Tareas discretas para implementación |

# Arquitectura de Dos Niveles para Prompts SDD

| Nivel | Ubicación | Propósito |
|---|---|---|
| **Nivel 1: Templates Globales** | `.specify/templates/` | Prompts reutilizables en cualquier proyecto |
| **Nivel 2: Prompts por Feature** | `specs/[feature]/` | Adaptaciones concretas para una feature específica |

# Arquitectura de Dos Niveles para Prompts SDD

| Nivel | Ubicación | Propósito |
|---|---|---|
| **Nivel 1: Templates Globales** | `.specify/templates/` | Prompts reutilizables en cualquier proyecto |
| **Nivel 2: Prompts por Feature** | `specs/[feature]/` | Adaptaciones concretas para una feature específica |

## Beneficios de Guardar Prompts

- **Trazabilidad:** Saber qué prompt generó cada artefacto
- **Reutilización:** Prompts probados en features similares
- **Colaboración:** Todo el equipo usa los mismos prompts
- **Auditoría:** Demonstrar cómo se generó el código

# SDD y Agile

El workflow SDD se integra con Scrum estructurándose en dos niveles temporales: Foundations y Desarrollo Iterativo.

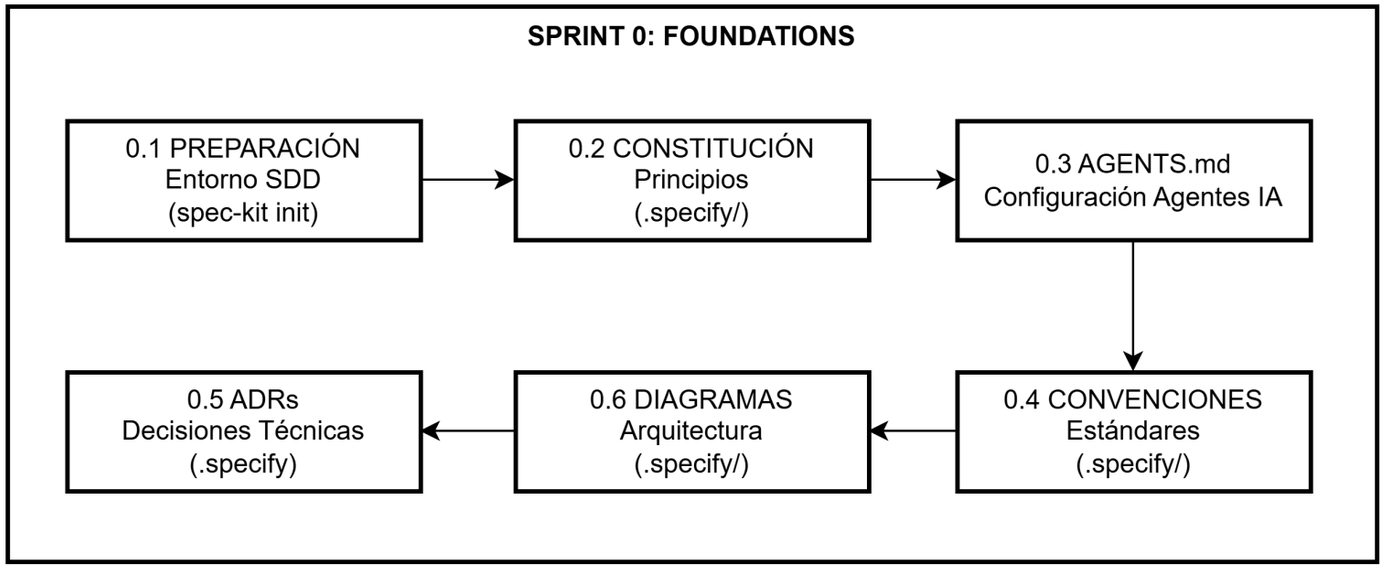
# 0: Foundations (Artefactos de Arquitectura)

# 0: Foundations (Artefactos de Arquitectura)

El Sprint 0 establece la base técnica del proyecto antes de comenzar el desarrollo iterativo. Durante este sprint se generan los artefactos fundamentales que guiarán todos los sprints posteriores:

# 0: Foundations (Artefactos de Arquitectura)

El Sprint 0 establece la base técnica del proyecto antes de comenzar el desarrollo iterativo. Durante este sprint se generan los artefactos fundamentales que guiarán todos los sprints posteriores:



# Artefactos del Sprint 0

| Paso | Artefacto | Descripción |
|---|---|---|
| 0.1 | Configuración SDD | `specify init`, AGENTS.md base |
| 0.2 | `constitution.md` | Principios tecnológicos, seguridad, calidad |
| 0.3 | *.agents.md | Configuración de agentes especializados |
| 0.4 | `conventions.md` | Estándares de código detallados |
| 0.5 | `adr.md` | Decisiones arquitectónicas documentadas |
| 0.6 | `architecture/diagrams.md` | Diagramas de arquitectura general |

# Sprints 1+: Desarrollo Iterativo

# Sprints 1+: Desarrollo Iterativo

Los sprints subsiguientes siguen el ciclo Scrum tradicional potenciado por SDD:

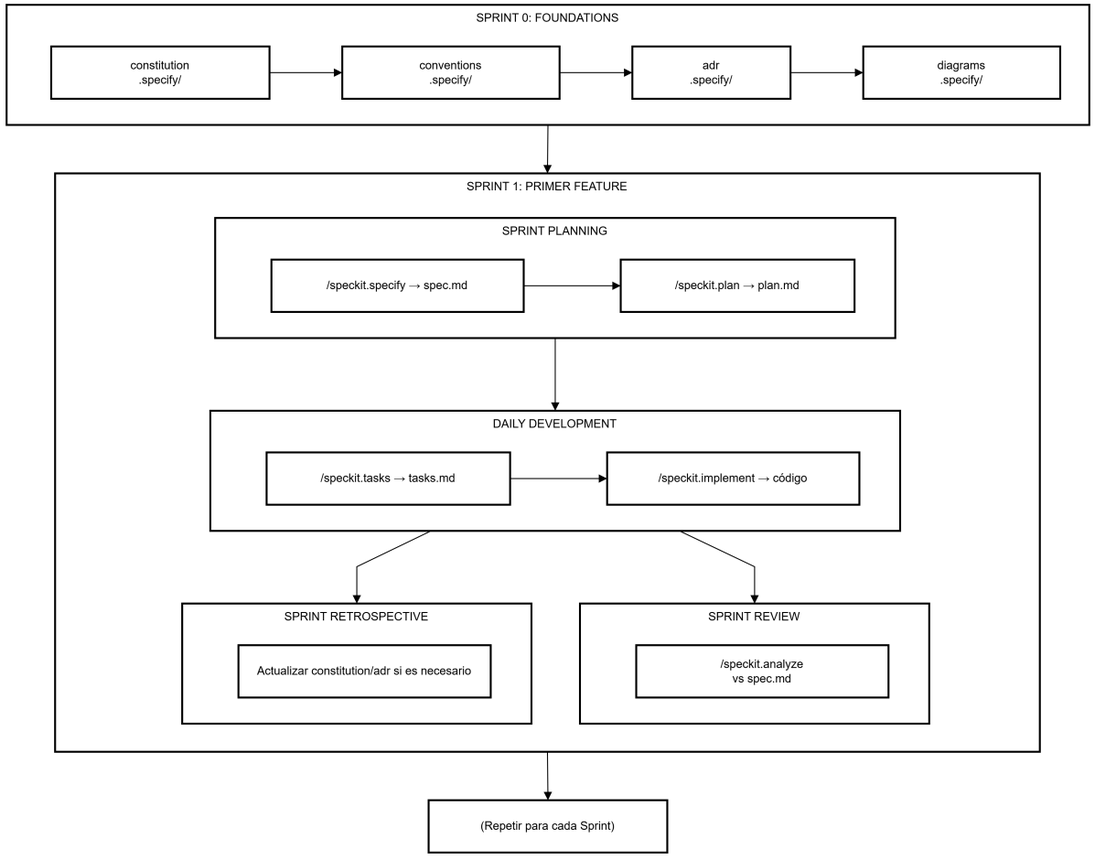
# Sprints 1+: Desarrollo Iterativo

Los sprints subsiguientes siguen el ciclo Scrum tradicional potenciado por SDD:

```
Sprint N (2 semanas)
├── Día 1: Sprint Planning
│   ├── /speckit.specify → Genera spec.md con user stories
│   └── /speckit.plan → Crea plan técnico
├── Días 2-9: Desarrollo
│   ├── /speckit.tasks → Desglosa en tareas por día
│   ├── Daily: Tarea(s) del día según tasks.md
│   └── /speckit.implement → Implementa con Copilot
├── Día 10: Sprint Review
│   ├── Comparar código generado vs spec.md
│   └── /speckit.analyze → Verificar cumplimiento
└── Día 10: Retrospective
    ├── Actualizar constitution.md si es necesario
    └── ¿Nuevo ADR necesario? → Documentar en adr.md
```

# Flujo Completo SDD + Scrum

# Flujo Completo SDD + Scrum



# Beneficios Clave

# Beneficios Clave

- **Contexto persistente:** Especificaciones, planes y tareas se almacenan como archivos markdown, creando un registro permanente

# Beneficios Clave

- **Contexto persistente:** Especificaciones, planes y tareas se almacenan como archivos markdown, creando un registro permanente
- **Trazabilidad completa:** Cada línea de código puede rastrearse a un requisito específico

# Beneficios Clave

- **Contexto persistente:** Especificaciones, planes y tareas se almacenan como archivos markdown, creando un registro permanente
- **Trazabilidad completa:** Cada línea de código puede rastrearse a un requisito específico
- **Colaboración mejorada:** Equipos comparten artefactos en `specs/001-feature/`

# Beneficios Clave

- **Contexto persistente:** Especificaciones, planes y tareas se almacenan como archivos markdown, creando un registro permanente
- **Trazabilidad completa:** Cada línea de código puede rastrearse a un requisito específico
- **Colaboración mejorada:** Equipos comparten artefactos en `specs/001-feature/`
- **Validación automatizada:** CI/CD puede verificar cumplimiento de especificaciones

# Beneficios Clave

- **Contexto persistente:** Especificaciones, planes y tareas se almacenan como archivos markdown, creando un registro permanente
- **Trazabilidad completa:** Cada línea de código puede rastrearse a un requisito específico
- **Colaboración mejorada:** Equipos comparten artefactos en `specs/001-feature/`
- **Validación automatizada:** CI/CD puede verificar cumplimiento de especificaciones
- **IA contextualizada:** Copilot siempre tiene contexto del proyecto vía AGENTS.md y skills

# Sprint 0

## Fase de Diseño y Arquitectura con SDD

# Introducción al Diseño SDD

# Introducción al Diseño SDD

- Las especificaciones de alto nivel se transforman en decisiones técnicas concretas

# Introducción al Diseño SDD

- Las especificaciones de alto nivel se transforman en decisiones técnicas concretas
- Vinculación estricta entre diseño y especificaciones previas

# Introducción al Diseño SDD

- Las especificaciones de alto nivel se transforman en decisiones técnicas concretas
- Vinculación estricta entre diseño y especificaciones previas
- Trazabilidad bidireccional entre decisiones y requisitos

# Introducción al Diseño SDD

- Las especificaciones de alto nivel se transforman en decisiones técnicas concretas
- Vinculación estricta entre diseño y especificaciones previas
- Trazabilidad bidireccional entre decisiones y requisitos
- Copilot como arquitecto asistente que genera propuestas

# Introducción al Diseño SDD

- Las especificaciones de alto nivel se transforman en decisiones técnicas concretas
- Vinculación estricta entre diseño y especificaciones previas
- Trazabilidad bidireccional entre decisiones y requisitos
- Copilot como arquitecto asistente que genera propuestas
- Integración con Spec Kit para flujo hacia implementación

# Introducción al Diseño SDD

- Las especificaciones de alto nivel se transforman en decisiones técnicas concretas
- Vinculación estricta entre diseño y especificaciones previas
- Trazabilidad bidireccional entre decisiones y requisitos
- Copilot como arquitecto asistente que genera propuestas
- Integración con Spec Kit para flujo hacia implementación

## Dónde guardar prompts de diseño:

`specs/[feature]/design-prompts.md` o directamente en `plan.md`

# Generación de Diagramas de Flujo

La generación de diagramas de procesos es el primer paso en la fase de diseño.

````
@architect Genera un diagrama de flujo Mermaid para el proceso de creación de tareas en PortalEmpleo:

## Especificación de Referencia
- Archivo: specs/001-task-api/spec.md
- Requisito: REQ-TASK-001 (Crear tarea)
- Criterios de aceptación: CA-TASK-001, CA-TASK-002

## Proceso de Negocio
1. Usuario autenticado accede a pantalla de creación
2. Sistema muestra formulario con campos
3. Usuario completa formulario y envía
4. Sistema valida datos
5. Sistema crea registro en base de datos
6. Sistema asigna ID único
7. Sistema retorna confirmación

## Formato de Salida
```mermaid
graph TD
````
````

# Diagrama de Arquitectura C4

```
@architect Genera diagrama C4 de arquitectura para PortalEmpleo basado en constitution.md:

## Contexto del Sistema
- PortalEmpleo: Sistema de gestión de tareas empresariales
- Usuarios: ~1000 concurrentes
- SLA: 99.5% disponibilidad, < 200ms respuesta

## Componentes Definidos en Specs
- API Gateway: Punto de entrada
- Auth Service: Autenticación JWT
- Task Service: Gestión de tareas
- User Service: Perfiles de usuario
- SQL Server: Base de datos principal
- Redis: Cache de sesiones

## Patrón Arquitectónico
- Microservicios con API Gateway
- Comunicación síncrona REST
- Base de datos por servicio
```

# Documentación de Decisiones Técnicas (ADR)

Los ADRs capturan decisiones arquitectónicas importantes y su justificación.

```
@architect un ADR para la decisión de arquitectura de datos en PortalEmpleo:

## Decisión a Documentar
- Usar Entity Framework Core con SQL Server como ORM primario
- Implementar Repository Pattern con Unit of Work

## Contexto
- Proyecto existente con legacy code en ADO.NET
- Equipo familiarizado con Entity Framework
- Requisito de migración gradual desde sistema legacy

## Opciones Consideradas
1. Continue with ADO.NET (status quo)
2. Dapper + repositories
3. Entity Framework Core + repositories

## Decisión Seleccionada
Entity Framework Core + repositories
```

# Validación de Arquitectura

La validación asegura que las decisiones cumplan con los requisitos no funcionales.

```
@architect Valida la siguiente propuesta de arquitectura contra constitution.md:

## Propuesta de Arquitectura
- API: REST sobre HTTPS (TLS 1.3)
- Database: SQL Server 2022 en Azure
- Auth: JWT con refresh tokens (30 min / 7 días)
- Deployment: Azure App Service con auto-scaling

## Requisitos No Funcionales de constitution.md
- APIs responden en < 200ms (p95)
- Todas las APIs requieren JWT de Microsoft Entra ID
- Datos cifrados in transit (TLS 1.2+) y at rest (AES-256)
- Audit log completo de cambios

## Matriz de Cumplimiento
| Requisito NF | Propuesta | Cumple | Gap |
|--------------|-----------|--------|-----|
| TLS 1.3 | TLS 1.3 | ✓ | - |
| JWT Entra ID | JWT custom | X | Migrar a Entra ID |
```

# Esqueletos de Proyectos C#

```
@architect Genera estructura completa de proyecto ASP.NET Core 8 para PortalEmpleo API:

## Requisitos Técnicos
- .NET 8, C# 12
- Clean Architecture
- Entity Framework Core 8
- SQL Server 2022
- xUnit para testing
- FluentValidation

## Estructura Requerida
src/
├── PortalEmpleo.API/
│   ├── Controllers/
│   ├── Middleware/
│   ├── Filters/
│   ├── Program.cs
│   └── appsettings.json
├── PortalEmpleo.Application/
```

# Program.cs Generado

```
using PortalEmpleo.Infrastructure.Data;
using PortalEmpleo.Infrastructure.Identity;
using PortalEmpleo.Application;
using PortalEmpleo.Infrastructure;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddApplication();
builder.Services.AddInfrastructure(builder.Configuration);
builder.Services.AddApiServices();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}

app.UseApiServices();
```

# Estructura de Notebooks Python

```
Genera estructura de proyecto Python para análisis de productividad:

## Stack Tecnológico
- Python 3.12
- Jupyter Notebooks
- Pandas, NumPy, Scikit-learn
- MLflow para tracking
- Plotly para visualización
- FastAPI para serving

## Estructura Requerida
notebooks/
├── 01-data-exploration.ipynb
├── 02-feature-engineering.ipynb
├── 03-model-training.ipynb
├── 04-model-evaluation.ipynb
├── 05-prediction-api.ipynb
└── 06-dashboard-analysis.ipynb
```

# Estructura React Native SDD

```
Genera estructura de proyecto React Native con Expo para PortalEmpleo Mobile:

## Stack Tecnológico
- React Native con Expo 51
- TypeScript 5
- React Query para data fetching
- React Navigation 6
- Zustand para state management

## Estructura Requerida
src/
├── components/
│   ├── common/
│   ├── forms/
│   └── layout/
├── screens/
│   ├── auth/
│   ├── tasks/
│   └── home/
```

# Generación Automática desde AGENTS.md

```
@agent Configura AGENTS.md para generación automática de esqueletos:

## Requerimiento
Cuando el agente detecte creación de nuevo feature,
deberá generar automáticamente la estructura completa.

## Patrón de Detección
- Directorio nuevo en specs/
- Nuevo spec.md con feature name
- Trigger: creación de primer archivo en src/[Feature]/

## Template a Usar
- Template: .specify/templates/feature-skeleton.md
- Ubicación: src/[FeatureName]/
- Archivos: controller, service, repository, dto, validator, tests
```

# Referencias Oficiales

Documentación y recursos adicionales

# Documentación de Herramientas

# Documentación de Herramientas

- **GitHub Copilot Documentation**: https://docs.github.com/en/copilot

# Documentación de Herramientas

- **GitHub Copilot Documentation**: https://docs.github.com/en/copilot
- **Spec Kit Repository**: https://github.com/github/spec-kit

# Documentación de Herramientas

- **GitHub Copilot Documentation**: https://docs.github.com/en/copilot
- **Spec Kit Repository**: https://github.com/github/spec-kit
- **Microsoft Learn - Spec-Driven Development**: https://learn.microsoft.com/es-es/training/modules/...

# Recursos de Patrones y Diagramas

# Recursos de Patrones y Diagramas

- **Entity Framework Core**: https://docs.microsoft.com/en-us/ef/core/

# Recursos de Patrones y Diagramas

- **Entity Framework Core**: https://docs.microsoft.com/en-us/ef/core/
- **MediatR**: https://github.com/jbogard/MediatR

# Recursos de Patrones y Diagramas

- **Entity Framework Core**: https://docs.microsoft.com/en-us/ef/core/
- **MediatR**: https://github.com/jbogard/MediatR
- **Clean Architecture**: https://github.com/ardalis/CleanArchitecture

# Recursos de Patrones y Diagramas

- **Entity Framework Core**: https://docs.microsoft.com/en-us/ef/core/
- **MediatR**: https://github.com/jbogard/MediatR
- **Clean Architecture**: https://github.com/ardalis/CleanArchitecture
- **Mermaid Chart Syntax**: https://mermaid.js.org/

# Recursos de Patrones y Diagramas

- **Entity Framework Core**: https://docs.microsoft.com/en-us/ef/core/
- **MediatR**: https://github.com/jbogard/MediatR
- **Clean Architecture**: https://github.com/ardalis/CleanArchitecture
- **Mermaid Chart Syntax**: https://mermaid.js.org/
- **C4 Model**: https://c4model.com/

# Recursos de Patrones y Diagramas

- **Entity Framework Core**: https://docs.microsoft.com/en-us/ef/core/
- **MediatR**: https://github.com/jbogard/MediatR
- **Clean Architecture**: https://github.com/ardalis/CleanArchitecture
- **Mermaid Chart Syntax**: https://mermaid.js.org/
- **C4 Model**: https://c4model.com/
- **ADR Documentation**: https://github.com/joelparkerhenderson/architecture-decision-record

# Resumen del Módulo 2

Conceptos clave y herramientas SDD

# Resumen: Conceptos y Herramientas

| Sección | Conceptos Clave | Herramientas/Artefactos |
|---|---|---|
| **2.1 Diseño SDD** | Diagramas C4, ADRs, esqueletos de proyecto | Spec Kit design prompts, templates de arquitectura |
| **2.2 Desarrollo SDD** | Generación de código con trazabilidad | tasks.md, code-prompts, patrones (Factory, Repository, Observer) |
| **Contexto SDD** | Arquitectura de dos niveles de prompts | Templates globales vs prompts por feature |

# Conceptos Clave del Módulo

# Conceptos Clave del Módulo

- Workflow Spec Kit completo: `specify` → `plan` → `tasks` → `implement` → `analyze`

# Conceptos Clave del Módulo

- Workflow Spec Kit completo: `specify` → `plan` → `tasks` → `implement` → `analyze`
- Arquitectura de dos niveles para prompts: templates globales y específicos por feature

# Conceptos Clave del Módulo

- Workflow Spec Kit completo: `specify` → `plan` → `tasks` → `implement` → `analyze`
- Arquitectura de dos niveles para prompts: templates globales y específicos por feature
- Trazabilidad completa desde specs hasta código implementado

# Conceptos Clave del Módulo

- Workflow Spec Kit completo: `specify` → `plan` → `tasks` → `implement` → `analyze`
- Arquitectura de dos niveles para prompts: templates globales y específicos por feature
- Trazabilidad completa desde specs hasta código implementado
- Patrones de diseño SDD: Factory, Repository, Observer, Domain Events

# Conceptos Clave del Módulo

- Workflow Spec Kit completo: `specify` → `plan` → `tasks` → `implement` → `analyze`
- Arquitectura de dos niveles para prompts: templates globales y específicos por feature
- Trazabilidad completa desde specs hasta código implementado
- Patrones de diseño SDD: Factory, Repository, Observer, Domain Events
- Integración con AGENTS.md para contexto continuo

# Conceptos Clave del Módulo

- Workflow Spec Kit completo: `specify` → `plan` → `tasks` → `implement` → `analyze`
- Arquitectura de dos niveles para prompts: templates globales y específicos por feature
- Trazabilidad completa desde specs hasta código implementado
- Patrones de diseño SDD: Factory, Repository, Observer, Domain Events
- Integración con AGENTS.md para contexto continuo

## Próximo paso:

El Módulo 3 profundiza en técnicas avanzadas de generación de código y desarrollo de APIs con Copilot.

# Gracias

## Módulo 2 Completado

Siguiente: Modulo 3 - Desarrollo de Código Potenciado por AI