

Documentación Inteligente de Proyectos con GitHub Copilot

Clase 2 - Análisis y Documentación Asistida por IA



¿Qué aprenderás hoy?



Configuración efectiva

Dominarás la configuración de GitHub Copilot para maximizar su potencial en documentación técnica



Técnicas de prompting

Aprenderás a crear prompts precisos que generen documentación de calidad profesional



Instructions y Prompt Files

Utilizarás herramientas estratégicas para mantener consistencia en todo el proyecto



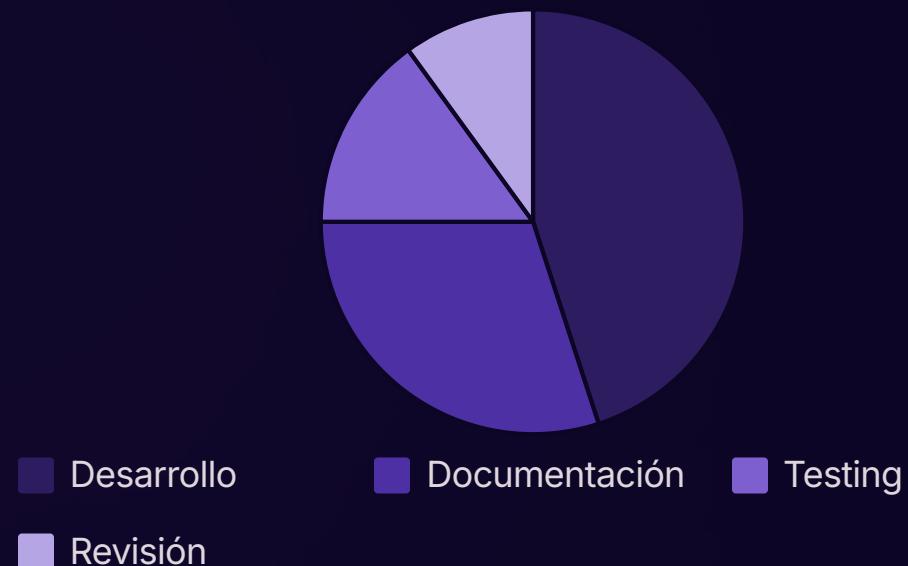
Modes contextuales

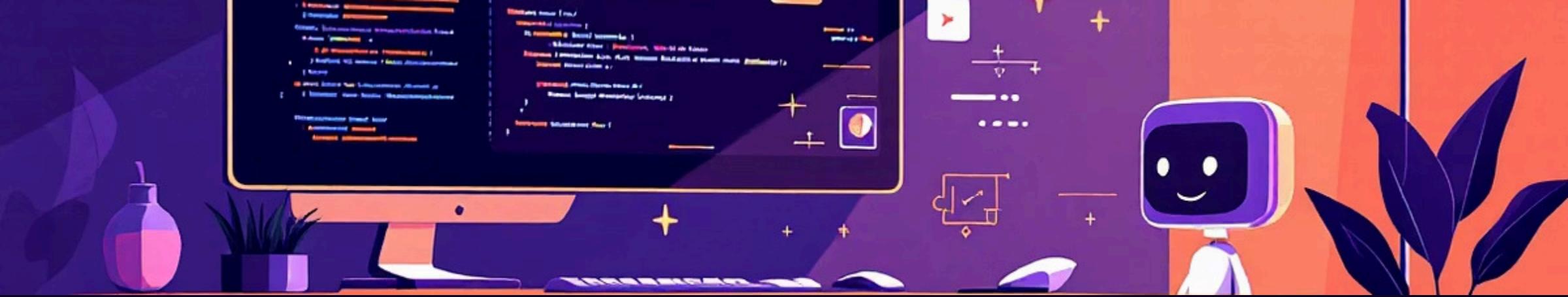
Aplicarás diferentes perspectivas de IA según el tipo de documentación requerida

El desafío de documentar código

¿Por qué documentar es difícil?

- Consume hasta el **30% del tiempo de desarrollo** en proyectos complejos
- La documentación desactualizada genera **deuda técnica** costosa
- Requiere cambios constantes de contexto mental entre código y explicación
- Diferentes audiencias necesitan distintos niveles de detalle técnico





GitHub Copilot como asistente de documentación

✓ Lo que SÍ hace

- **Analiza código** y extrae la intención del desarrollador automáticamente
- **Genera documentación** en múltiples formatos: Markdown, Gherkin, JSDoc, etc.
- **Mantiene consistencia** de estilo según las convenciones del proyecto
- **Acelera la escritura** técnica reduciendo el tiempo de documentación hasta un 60%

□ Lo que NO hace

- Reemplazar el **juicio técnico humano** en decisiones críticas
- Entender automáticamente el **contexto de negocio** sin información adicional
- Validar la **exactitud técnica** sin supervisión del desarrollador
- Capturar requisitos no funcionales o restricciones del dominio

Instructions: El contexto permanente

¿Qué son las Instructions?

Archivo de configuración (.github/copilot-instructions.md) que Copilot lee automáticamente en cada interacción.

Define las reglas base del proyecto para todo el equipo y se versiona con Git.

¿Cuándo usarlas?

- Convenciones de nomenclatura específicas del proyecto
- Patrones arquitectónicos y estándares de diseño
- Formatos corporativos de documentación técnica
- Contexto del dominio de negocio (ERP, finanzas, etc.)

```
.github/  
└── copilot-instructions.md  
└── Contexto del proyecto  
└── Convenciones de código  
└── Formatos de documentación
```

Prompt Files: Templates reutilizables

¿Qué son los Prompt Files?

Archivos .md con **prompts predefinidos** y plantillas estructuradas que se guardan en el repositorio. Son reutilizables por todo el equipo y mantienen la consistencia.

¿Cuándo usarlos?

- Templates de documentación: Gherkin, ADR, RFC
- Checklists de revisión de código estandarizadas
- Guías de análisis técnico recurrentes
- Formatos corporativos para diferentes tipos de entregables

```
.copilot/
  └── templates/
    ├── gherkin-template.md
    ├── test-template.md
    └── review-checklist.md
  └── prompts/
    └── analysis-prompt.md
```

- **Ventaja clave:** Los Prompt Files se versionan con el código, garantizando que todo el equipo use las mismas plantillas actualizadas.

"Modes": Cambiando el rol de Copilot

Los **modes** son una técnica avanzada de prompting que cambia la perspectiva de Copilot dinámicamente en cada conversación, complementando Instructions y Prompt Files.

Documentation Mode

Rol: Business Analyst / Technical Writer

Objetivo: Documentar funcionalidad para usuarios finales y stakeholders

Salida: Gherkin, diagramas de flujo, manuales de usuario

Testing Mode

Rol: QA Engineer / Test Architect

Objetivo: Diseñar estrategia completa de pruebas

Salida: Casos de prueba, tests unitarios, matrices de cobertura

Review Mode

Rol: Senior Architect / Code Reviewer

Objetivo: Identificar mejoras y vulnerabilidades

Salida: Análisis crítico, recomendaciones arquitectónicas

¿Cuándo usar cada herramienta?

Característica	Instructions	Prompt Files	Modes
Persistencia	Permanente	Permanente	Temporal
Activación	Automática	Manual (referencia)	Mode (chat)
Alcance	Todo el proyecto	Tareas específicas	Tareas especializadas
Flexibilidad	Reglas fijas	Templates adaptables	Contexto dinámico
Colaboración	Todo el equipo	Todo el equipo	Todo el equipo
Ejemplo	"Usa español"	"Formato Gherkin"	"Actúa como BA"

La **clave del éxito** está en combinar las tres herramientas de forma estratégica según el contexto de cada tarea de documentación.

Workflow integrado: Combinando las tres herramientas



Instructions

Contexto base del proyecto: lenguaje, convenciones, patrones arquitectónicos



Prompt File

Template o estructura específica para el tipo de documentación requerida



Mode

Perspectiva y enfoque según la audiencia: técnica, negocio o QA



Resultado

Documentación perfecta: consistente, completa y alineada con el proyecto

- Ejemplo práctico:** Instructions define "Proyecto D365, español, patrón ttsBegin/Commit" + Prompt File aporta "Template Gherkin" + Mode aplica "Business Analyst" = Documentación Gherkin en español, alineada con D365, perspectiva de negocio.



Fórmula para prompts efectivos

01

Rol/Mode

"Actúa como [Technical Writer/BA/Architect]"

02

Contexto

"@workspace #file:archivo.x Analiza este código de [dominio]"

03

Tarea específica

"Genera [tipo de documentación] que explique [aspecto]"

04

Formato

"Usa [template/estructura] y [convenciones del proyecto]"

Ejemplo completo: "Actúa como Business Analyst especializado en ERP. @workspace #file:PWProdNotificar.xpo Genera documentación en formato Gherkin que explique el proceso de notificación de producción. Usa el template de .copilot/gherkin-template.md y las convenciones en español del proyecto."