

Desarrollo asistido por IA, documentación preparada para MCP y entrega del proyecto final

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Disciplina en la programación asistida por IA, acceso a la documentación, prompts de revisión, defensa empresarial final. Lo que esto significa para un equipo avanzado de Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Disciplina en la programación asistida por IA, acceso a la documentación, prompts de revisión, defensa final del proyecto empresarial.

1. El problema en producción

Los equipos avanzados usarán herramientas de IA, pero el trabajo empresarial con Wisej.NET sigue requiriendo arquitectura, documentación basada en las fuentes, revisión de código, revisión de seguridad y responsables operativos.

2. Conceptos clave

Concepto	Por qué importa
desarrollo asistido por IA	Usar herramientas de IA para redactar borradores de código, pruebas y documentación, mientras el equipo sigue siendo responsable de revisar, fundamentar y asumir el resultado.
prompts basados en las fuentes	Prompts que dirigen al asistente hacia el código, la documentación y las convenciones reales del proyecto, para que el resultado generado coincida con la arquitectura real.
endpoint de documentación MCP	Un servidor Model Context Protocol que expone la documentación del equipo a las herramientas de IA, para que las respuestas procedan de las fuentes del propio proyecto.
lista de comprobación de revisión	Una lista fija de comprobaciones de arquitectura, seguridad y convenciones que se aplica a cada cambio generado por IA antes de aceptarlo.
defensa de la arquitectura	Presentar y justificar con pruebas las decisiones de diseño del proyecto final, para que el equipo pueda cuestionarlas y registrar el resultado.
demo del proyecto final	La demostración de principio a fin de la aplicación terminada, que prueba que los patrones de los módulos funcionan juntos en condiciones realistas.

3. Desarrollo de la lección

La IA ayuda al equipo a avanzar más rápido

La IA puede redactar borradores de pantallas, servicios, pruebas, prompts, documentación e inventarios de migración. No puede hacerse cargo del riesgo en producción. El desarrollador avanzado debe aportar el contexto de arquitectura, verificar las API, revisar el código generado y probar las rutas de fallo.

Usa la documentación como fuente para las herramientas

La documentación de Wisej.NET puede tratarse como fuente de verdad sobre el comportamiento actual de la plataforma. Los prompts de IA deberían pedir código que siga las convenciones del proyecto y exigir citas o referencias cuando se usen API específicas de la plataforma.

Velocidad sin código sin revisar

La asistencia de la IA es más valiosa en los márgenes del trabajo: redactar el esqueleto de un servicio a partir de la definición de un comando, generar casos de prueba para una matriz de rutas de fallo, escribir un inventario de migración a partir de un archivo de solución o convertir un runbook en una lista de comprobación. En todos esos casos el desarrollador ya sabe cómo es el resultado correcto y puede verificar la salida rápidamente. Es menos valiosa donde la verificación es difícil, como la lógica de seguridad, la concurrencia o las API de la plataforma que el desarrollador no conoce bien.

La regla que aplica el laboratorio es que el código generado es un borrador de un colaborador desconocido. Pasa por la misma revisión que cualquier pull request, con las mismas preguntas: dónde vive el estado, qué puede fallar, qué es seguro y qué cambia en producción. La ruta de fallo del laboratorio muestra por qué. Un servicio generado se inventa una API de Wisej.NET que no existe y guarda datos por usuario en un campo estático. Ambas cosas compilarían en una demo rápida y ambas fallarían con sesiones reales.

Acepta una sugerencia solo cuando alguien del equipo pueda explicarla sin tener la herramienta delante.

Prompts y listas de comprobación basados en el proyecto

La calidad del código generado depende de la calidad del contexto que se le ha dado. Cada sesión de IA del laboratorio empieza con una cabecera de prompt que enuncia las reglas del proyecto: la estructura de la solución, dónde vive la lógica de negocio, que los estáticos nunca contienen datos de sesión, que los servicios aplican los permisos y que solo se pueden usar API de Wisej.NET documentadas, con una referencia para cada una. La biblioteca de prompts reúne estas cabeceras y los prompts específicos de cada tarea para que todo el equipo parta de la misma base.

La lista de comprobación de revisión del código generado es la otra mitad. Es lo bastante corta como para aplicarla a cada cambio y lo bastante específica como para detectar los fallos que ha enseñado el curso: un campo estático que guarda estado de usuario o de tenant, un manejador de la UI que contiene lógica de persistencia, una API que no aparece en la documentación, una comprobación de permisos que falta en un servicio, una excepción mostrada al usuario y un recurso que nunca se libera.

- ¿Nombra el prompt las convenciones que debe seguir el resultado?
- ¿Cita la documentación cada afirmación específica de la plataforma que aparece en el resultado?
- ¿Ha aplicado y firmado la lista de comprobación un revisor, y no el autor?

Documentación que pueden leer las herramientas y los mantenedores

Un proyecto que se mantendrá con asistencia de IA necesita documentación que una herramienta pueda recuperar, no solo que una persona pueda consultar. Un endpoint de documentación MCP expone al asistente las notas de arquitectura, los registros de decisiones, las convenciones y los runbooks del propio equipo, de modo que las respuestas sobre este proyecto proceden de sus fuentes y no de datos de entrenamiento generales. El mismo endpoint sirve a un nuevo miembro del equipo que haga las mismas preguntas.

Eso solo funciona si la documentación está estructurada. El índice de documentación del laboratorio enumera cada documento con una línea sobre su propósito, las decisiones que recoge y la fecha de su última verificación. Los registros de decisiones de arquitectura del Módulo 1 son las entradas más valiosas, porque explican por qué existe un diseño y una herramienta puede citarlos cuando se le pide cambiarlo. Les siguen los runbooks, la matriz de permisos, la matriz de rutas de fallo y la tabla de configuración.

Escribe para un lector que nunca ha conocido al equipo. Si una nota depende de un conocimiento que solo existe en la cabeza de alguien, dentro de un año será incorrecta.

Defender el proyecto final como una revisión de arquitectura

El proyecto final se entrega como una revisión de arquitectura de producción, no como una demo de funcionalidades. El guion de la demo guía a un revisor por el EnterpriseOps Command Center en un orden fijo: la estructura de la solución y los registros de decisiones, una sesión consciente del tenant y una solicitud entre tenants rechazada, un comando que viaja a través de un servicio hasta la base de datos y vuelve con su entrada de auditoría, una importación en segundo plano con progreso y cancelación, un flujo de trabajo con un fallo compensado, la página de diagnóstico y el paquete de despliegue con su reversión.

Cada parada combina una pantalla que funciona con una ruta de fallo, porque la revisión trata sobre lo que ocurre cuando las cosas van mal. El revisor preguntará dónde vive el estado, quién puede llamar a qué, cómo se revierte una versión defectuosa y qué haría el equipo a las dos de la madrugada. La declaración de preparación para producción responde a todo eso por escrito, enumera lo que aún no está hecho y nombra al responsable de cada tarea pendiente.

Una defensa tiene éxito cuando cada decisión se puede vincular a un motivo, cada riesgo tiene una mitigación con nombre y el equipo puede decir con claridad qué cambiaría a continuación.



Regla avanzada de producción

No te detengas cuando la pantalla parezca correcta. Explica a quién pertenece el estado, las implicaciones de seguridad, las rutas de fallo y el comportamiento en producción antes de dar el módulo por terminado.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Usar la asistencia de IA para acelerar el desarrollo con Wisej.NET sin aceptar código sin revisar.

- ✓ Crear prompts y listas de comprobación de revisión basados en los estándares del proyecto y en la documentación oficial.
- ✓ Preparar notas de proyecto listas para MCP y documentación destinadas a los futuros mantenedores.
- ✓ Entregar y defender el proyecto final avanzado como una revisión de arquitectura de producción.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Completa el paquete del proyecto final. Añade notas sobre el uso de la IA, una biblioteca de prompts, enlaces a la documentación, una lista de comprobación de revisión de código, el guion final de la demo, el esquema de la presentación de defensa de la arquitectura y una declaración de preparación para producción.

Enfoque sugerido

- ✓ Lee el problema en producción e identifica el riesgo de arquitectura antes de tocar la UI.
- ✓ Crea o actualiza primero el límite correspondiente de servicio, comando, modelo o componente.
- ✓ Construye la pantalla o el componente Wisej.NET siguiendo los estándares de proyecto establecidos.
- ✓ Añade una demostración de la ruta de fallo, no solo del caso ideal.
- ✓ Anota lo que ha cambiado en el diario de aprendizaje y actualiza la lista de comprobación correspondiente.

Entregables requeridos

- ✓ Biblioteca de prompts de IA
- ✓ Lista de comprobación de revisión del código generado
- ✓ Índice de la documentación del proyecto
- ✓ Guion de la demo del proyecto final
- ✓ Declaración de preparación para producción

Preguntas de repaso

- ✓ ¿Qué código generado se aceptó y por qué?
- ✓ ¿Se comprobó con la documentación cada afirmación específica de la plataforma?
- ✓ ¿Puede el estudiante defender las decisiones de despliegue, seguridad y estado?



Regla avanzada de producción

No te detengas cuando la pantalla parezca correcta. Explica a quién pertenece el estado, las implicaciones de seguridad, las rutas de fallo y el comportamiento en producción antes de dar el módulo por terminado.