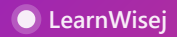


Ejercicio de programación con IA

Construye el ejemplo FieldOps de este módulo con ChatGPT o Claude a partir de un prompt listo para usar que dirige al modelo a la especificación del módulo, y después revisa, ejecuta y amplía lo que te devuelva.



No vas a teclear este ejemplo. Vas a dirigirlo.

Ya casi nadie escribe código Wisej.NET partiendo de un archivo vacío — un asistente escribe el primer borrador y el desarrollador decide si es correcto. Eso solo funciona si dominas el tema, y eso es lo que te dio el resto de este módulo. Aquí le pasas ese conocimiento a ChatGPT o a Claude, recibes el ejemplo de este módulo como una solución de Visual Studio que funciona y lo revisas como revisarías el pull request de un compañero.

Qué estás pidiendo

Crea la solución FieldOps con una lista de órdenes de trabajo, una página de detalle de la orden de trabajo y un flujo de actualización del estado que funcionen en un navegador normal. Redacta el registro de decisión de distribución: compara el navegador, la PWA, el shell de escritorio y el shell móvil según las necesidades de los técnicos, elige los destinos y su orden, y define las comprobaciones de capacidades que la aplicación usará para activar las funciones del dispositivo.

El módulo pide estos entregables, y el prompt de abajo le pide al asistente exactamente esos:

- ✓ Solución FieldOps con lista, detalle y actualización del estado
- ✓ Tabla comparativa de los modelos de distribución
- ✓ Registro de decisión de distribución con el orden de los destinos
- ✓ Diseño de la comprobación de capacidades
- ✓ Diagrama de arquitectura con un servidor y varios shells

Por qué tiene esta forma

FieldOps es una aplicación de servicio técnico de campo: los técnicos reciben órdenes de trabajo, capturan fotos y firmas, se desplazan hasta los emplazamientos y trabajan con una conectividad irregular.

Construida como una aplicación Wisej.NET normal ya funciona en cualquier navegador, lo que basta para los operadores de la oficina y no basta para un técnico que está en una sala de máquinas con una sola barra de cobertura y el teléfono en una mano con guante. La pregunta que responde este módulo no es «¿podemos construirla?», sino «¿cómo llega a las personas que la usan?», y esa respuesta es una decisión de arquitectura, no un detalle de empaquetado para el final.

Estos son los términos que el asistente tiene que dominar. Si lo que te devuelve los usa a la ligera, no ha entendido el módulo:

Término	Qué significa aquí
modelo de distribución	La vía por la que la aplicación llega a un usuario: una pestaña del navegador, una PWA instalada, una ventana de escritorio o una aplicación móvil distribuida por una tienda. Cambia la distribución, las actualizaciones y el acceso al dispositivo, nunca el código C# que construye la pantalla.
motor de navegador anfitrión	Todos los modelos renderizan la misma interfaz controlada por el servidor en un motor de navegador: el navegador del propio usuario o uno integrado (WebView2 en Windows, WebKit en macOS e iOS, el WebView del sistema en Android).
Wisej.NET Hybrid	El shell del framework basado en .NET MAUI que aloja tu aplicación Wisej.NET existente en Android, iOS y escritorio, y añade un puente hacia funciones del dispositivo como la cámara, la geolocalización, los archivos y las notificaciones.
puente con el dispositivo	El canal bidireccional entre tu código C# del servidor y el shell nativo: el servidor pide una acción del dispositivo, el shell la realiza y el resultado vuelve a tu código como un evento, no como un valor de retorno.
comprobación de capacidades	Una prueba en tiempo de ejecución de lo que el shell actual puede hacer realmente, que se usa para mostrar, ocultar o sustituir una función para que una misma pantalla se comporte correctamente en todos los modelos de distribución.
caché del shell de la aplicación	El HTML, el CSS, los scripts y los iconos estáticos que el service worker de una PWA puede conservar localmente. Es lo que sigue cargándose sin red; la parte interactiva de la aplicación, renderizada por el servidor, no está en ella.
asimetría de actualización	El servidor puede volver a desplegarse en cualquier momento y todas las sesiones reciben la nueva compilación al recargar, mientras que un instalador de escritorio o un paquete de tienda necesitan antes una publicación y una instalación.

Constrúyelo — elige tu asistente

Los dos botones abren una pestaña nueva con el **mismo prompt listo para usar** ya escrito. El prompt dirige al modelo a la especificación de este módulo — un único archivo Markdown con la lección, el enunciado del laboratorio y los entregables — para que trabaje con el material real en lugar de adivinar de qué trata el Módulo 1. Puedes abrir tú mismo [module-01.md](#) para ver exactamente lo que leerá.



Usar ChatGPT

Abre [chatgpt.com](#) con el prompt de este módulo ya cargado. Funciona con GPT-5 o Codex; cualquier plan que pueda navegar por la web descargará la especificación.



Usar Claude

Abre [claude.ai](#) con el mismo prompt ya cargado. Si usas Claude Code, puedes pegarlo en una terminal abierta en una carpeta vacía.



Envíalo tal cual y obtendrás un proyecto, no un fragmento

El prompt termina pidiendo toda la solución como un único `.zip` con los pasos para compilarla y ejecutarla. Descárgalo, abre la solución en Visual Studio, pulsa F5 y FieldOps se abrirá en `localhost`. Necesitas el SDK de .NET y el paquete NuGet `Wisej-4`; todo lo demás, el prompt le pide al asistente que lo escriba él mismo.

Estoy siguiendo el Módulo 1 – «Modelos de alojamiento más allá del navegador» – del curso de LearnWisej «Empaquetado híbrido y nativo», y quiero construir contigo el ejemplo de este módulo. Respóndeme en español.

Empieza leyendo la especificación del módulo. Es un único archivo Markdown (en inglés) que contiene la lección completa, el enunciado del laboratorio y los entregables:

<https://learnwisej.com/assets/courses/hybrid-native-packaging/specs/module-01.md>

Después construye un proyecto Wisej.NET 4 completo que pueda abrir en Visual Studio:

1. Proyecto «FieldOps», C#, net8.0 o posterior, paquete NuGet Wisej-4 4.1.x, host Kestrel que llame a `app.UseWisej()`.
2. Construye lo que pide el laboratorio de la especificación: Crea la solución FieldOps con una lista de órdenes de trabajo, una página de detalle de la orden de trabajo y un flujo de actualización del estado que funcionen en un navegador normal. Redacta el registro de decisión de distribución: compara el navegador, la PWA, el shell de escritorio y el shell móvil según las necesidades de los técnicos, elige los destinos y su orden, y define las comprobaciones de capacidades que la aplicación usará para activar las funciones del dispositivo.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Solución FieldOps con lista, detalle y actualización del estado
 - Tabla comparativa de los modelos de distribución
 - Registro de decisión de distribución con el orden de los destinos
 - Diseño de la comprobación de capacidades
 - Diagrama de arquitectura con un servidor y varios shells
4. Sigue la especificación siempre que nombre una clase, propiedad o evento de Wisej.NET – usa exactamente ese. Donde deje la elección abierta, dime qué API elegiste y por qué.
5. Mantén la autoridad en el servidor: el estado y la validación viven en .NET, y el navegador muestra e informa. Nunca traslades una decisión a JavaScript porque allí resulte más fácil.
6. Haz que el resultado se pueda demostrar: debo poder ejecutarlo y ver el comportamiento del que trata este módulo, incluido lo que ocurre cuando algo sale mal.

Antes de escribir una sola línea de código, explica con tus propias palabras qué vas a construir y dónde vivirá cada pieza, para que pueda corregir el diseño mientras todavía es barato.

Cuando el código esté terminado, empaqueta toda la solución en un único archivo .zip descargable y dame los pasos exactos para compilarla y ejecutarla.

Por último, responde a estas preguntas de la especificación usando tu propia implementación como prueba:

- Un técnico pierde la señal a mitad de la actualización de una orden de trabajo. Describe exactamente qué partes de FieldOps se pueden seguir mostrando desde un shell en caché, cuáles no, y qué debería hacer tu pantalla en el momento en que se cae la conexión.
- Tu registro de decisión coloca el shell de Android antes que la PWA. ¿Qué tendría que ser cierto sobre los dispositivos y la red de los técnicos para que ese orden fuera incorrecto?
- La página de detalle necesita una foto. Explica por qué una comprobación de capacidades con una alternativa Upload es un diseño mejor que un interruptor en tiempo de compilación que genere una versión de la pantalla solo para móvil.

Pregúntame por cualquier cosa ambigua antes de empezar.

Después revisa lo que te devuelve

Que el código compile no significa que sea correcto. Las preguntas interesantes son de diseño: ¿es este el control de Wisej.NET que la tarea realmente necesita?, ¿sigue el ciclo de vida que te ofrece el framework?, ¿el estado de referencia está en el servidor o se quedó en el navegador porque era más cómodo? Lee el resultado como revisor, no como cliente.

Este módulo te da la lista de comprobación. Responde a cada punto basándote en el código que acabas de recibir:

- ✓ Un técnico pierde la señal a mitad de la actualización de una orden de trabajo. Describe exactamente qué partes de FieldOps se pueden seguir mostrando desde un shell en caché, cuáles no, y qué debería hacer tu pantalla en el momento en que se cae la conexión.
- ✓ Tu registro de decisión coloca el shell de Android antes que la PWA. ¿Qué tendría que ser cierto sobre los dispositivos y la red de los técnicos para que ese orden fuera incorrecto?
- ✓ La página de detalle necesita una foto. Explica por qué una comprobación de capacidades con una alternativa Upload es un diseño mejor que un interruptor en tiempo de compilación que genere una versión de la pantalla solo para móvil.

Todo lo que hayas tenido que corregir merece quedar por escrito. Una nota breve con las convenciones y decisiones que acordaste, pegada en el primer mensaje de tu próxima sesión, vale más que cualquier prompt ingenioso.



Todavía no hay un ejemplo de referencia para este curso — el tuyo es el primero

El [repositorio de ejemplos de LearnWisej](#) tiene soluciones completas para la mayoría de los cursos, pero todavía no para este. Eso significa que no hay nada con qué compararte ni ningún atajo: la especificación y tu propia revisión son toda tu red de seguridad. Guarda lo que construyas — y si sale bien, nos encantaría verlo.

Tres prompts para ir más allá

Dirige tu asistente al proyecto que acabas de construir y prueba estos. Cada uno es un ejercicio de revisión y ampliación, no una reescritura.

- ✓ **Cambia una pieza sin mover las juntas.** *"Elige la parte de este proyecto FieldOps que sea más claramente un sustituto de algo real — una fuente de datos falsa, un almacén en memoria, un servicio simplificado — y reemplázala por una implementación con forma de producción. El contrato a ambos lados no debe cambiar: los mismos tipos, las mismas llamadas, el mismo comportamiento ante fallos. Dime qué te obligó a gestionar la implementación real que el sustituto no gestionaba."*
- ✓ **Haz que aguante un mal día.** *"Revisa este proyecto FieldOps en busca de los modos de fallo que todavía no gestiona — una entrada incorrecta o ausente, una dependencia lenta o no disponible, trabajo que llega después de que el usuario haya salido de la pantalla y dos acciones que compiten"*

entre sí. Para cada uno, dime primero si es un riesgo real en Wisej.NET 4 y por qué, y solo después escribe la corrección. Si decides que algo no necesita gestión, dilo y explica el razonamiento en lugar de añadir código defensivo."

- ✓ **Añade el segundo caso.** "Amplía este proyecto FieldOps con un segundo caso que siga el mismo patrón que el primero — otra entidad, pantalla o control con la misma forma, conectado de la misma manera. Extrae a una clase base o a un servicio lo que de verdad sea común, pero solo donde se justifique; dos cosas parecidas no siempre piden una abstracción. Muéstrame el diff y justifica cada cosa que hayas extraído."



Llévalo al laboratorio

El laboratorio práctico que cierra este módulo pide los mismos entregables que acabas de hacer que generara el asistente. Usa lo que tienes aquí como forma de tu respuesta y como prueba que la respalda — no como la respuesta en sí. El laboratorio se califica por tu razonamiento, y esa parte tiene que ser tuya.