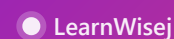


Ejercicio de programación con IA

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.



El mismo módulo, construido como de verdad vas a construir.

Todas las demás secciones de este módulo explican cómo funciona «API del dispositivo a través del puente híbrido». Esta te pide que lo generes — dirigiendo a un asistente con la propia especificación del módulo y comprobando después que el resultado se ejecuta y hace lo que describe el enunciado del laboratorio. Es el flujo de trabajo que usarás en proyectos reales, practicado con algo cuya respuesta correcta ya conoces.

Qué estás pidiendo

Añade funciones del dispositivo a FieldOps: una acción Take Photo que use la cámara del dispositivo en el shell nativo y recurra a un control Upload en el navegador, una captura de firma subida como imagen, una acción Navigate que lea la ubicación del dispositivo y abra la aplicación de mapas, una notificación push cuando se asigne una orden de trabajo, y una comprobación de capacidades que muestre u oculte cada acción según el shell.

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

- ✓ Captura con la cámara con alternativa de subida en el navegador
- ✓ Captura y subida de la firma
- ✓ Geolocalización y traspaso a la aplicación de mapas
- ✓ Notificación push al asignar
- ✓ Matriz de comprobación de capacidades entre shells

Por qué tiene esta forma

Un técnico de FieldOps fotografía una válvula corroída, recoge una firma sobre la pantalla y conduce hasta la siguiente dirección, mientras el operador que le asignó el trabajo sigue la misma orden de trabajo en un navegador de escritorio. Cada control de FieldOps es un objeto C# en tu servidor, que no puede tocar la cámara, el chip GPS ni la bandeja de notificaciones. Este módulo es el viaje de ida y vuelta que cierra esa brecha, y lo que hace la pantalla donde ese viaje no existe.

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
puente con el dispositivo	El canal de mensajes que un shell híbrido expone a la página que aloja: tu servidor pide al shell que realice una operación del dispositivo, y el shell responde con un mensaje que la aplicación convierte de nuevo en un evento del servidor.
objeto host	El objeto que un WebView inyecta en la página alojada para que el JavaScript de esa página pueda llamar a código nativo. El código C# del servidor solo llega a él mediante la interoperabilidad con JavaScript de la aplicación.
descriptor de capacidades	Las funciones que comunica un shell al iniciar una sesión (cámara, ubicación, notificaciones, biometría), guardadas y consultadas antes de mostrar cualquier acción del dispositivo.
mejora progresiva	Diseñar cada pantalla para que la vía del navegador normal sea una función completa y la vía nativa sea una versión mejor de ella, nunca la única que funciona.
permiso en el momento justo	Pedir a la plataforma acceso a la cámara, la ubicación o las notificaciones en el momento en que el usuario pulsa el botón que lo necesita, justo después de una frase que explica por qué.
evento de resultado del puente	La respuesta asíncrona (bytes de la foto, coordenadas o una negativa) que llega en una solicitud posterior al clic que la pidió, por lo que necesita su propio envío al navegador.
almacenamiento fuera de la raíz web	Escribir las fotos y firmas capturadas donde ninguna URL apunta a ellas, y servir las después mediante un controlador que comprueba si esta sesión puede ver esa orden de trabajo.

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 5. Puedes abrir tú mismo [module-05.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 5 – «API del dispositivo a través del puente híbrido» – 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-05.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: Añade funciones del dispositivo a FieldOps: una acción Take Photo que use la cámara del dispositivo en el shell nativo y recurra a un control Upload en el navegador, una captura de firma subida como imagen, una acción Navigate que lea la ubicación del dispositivo y abra la aplicación de mapas, una notificación push cuando se asigne una orden de trabajo, y una comprobación de capacidades que muestre u oculte cada acción según el shell.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Captura con la cámara con alternativa de subida en el navegador
 - Captura y subida de la firma
 - Geolocalización y traspaso a la aplicación de mapas
 - Notificación push al asignar
 - Matriz de comprobación de capacidades entre 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:

- El resultado de la cámara llega cuarenta segundos después del clic que lo pidió. Explica en qué solicitud llega, qué estado de sesión debe encontrar para saber qué hacer con los bytes y por qué el cuadro de imagen no aparece en el navegador sin un envío explícito.
- Hace tres semanas, un técnico denegó de forma permanente el permiso de ubicación en su propio teléfono. Describe exactamente qué debería hacer hoy la pulsación de Navigate y nombra lo único que la aplicación no debe hacer nunca en respuesta.
- Take Photo funciona en el shell móvil y la alternativa Upload funciona en Chrome de escritorio. Nombra algo que sea realmente distinto para el usuario entre esas dos vías y argumenta si esa diferencia es aceptable para un técnico de campo.

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:

- ✓ El resultado de la cámara llega cuarenta segundos después del clic que lo pidió. Explica en qué solicitud llega, qué estado de sesión debe encontrar para saber qué hacer con los bytes y por qué el cuadro de imagen no aparece en el navegador sin un envío explícito.
- ✓ Hace tres semanas, un técnico denegó de forma permanente el permiso de ubicación en su propio teléfono. Describe exactamente qué debería hacer hoy la pulsación de Navigate y nombra lo único que la aplicación no debe hacer nunca en respuesta.
- ✓ Take Photo funciona en el shell móvil y la alternativa Upload funciona en Chrome de escritorio. Nombra algo que sea realmente distinto para el usuario entre esas dos vías y argumenta si esa diferencia es aceptable para un técnico de campo.

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.

- ✓ **Revísalo como un pull request.** *"Explícame este código de FieldOps como si estuvieras revisando el pull request de un compañero y no lo hubieras escrito tú. ¿Dónde vive el estado?, ¿qué ocurre de principio a fin con una acción del usuario?, ¿qué devolverías para que lo cambien? y ¿qué partes no te sentirías cómodo poniendo delante de un cliente? Sé concreto y cita las líneas a las que te refieres — no me resumas el diseño."*
- ✓ **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."*



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.