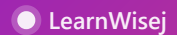


Ejercicio de programación con IA

Construye el ejemplo OpsMonitor 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 — incluso comparándolo con nuestra propia solución de referencia en GitHub.



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 WisejPerfLab con un dashboard de soporte, un grid de tickets y un árbol de clientes alimentados con datos de prueba, y hazla medible. Añade un servicio ScenarioProbe que envuelva un escenario en un ámbito Stopwatch y escriba logs PERF estructurados de inicio y fin con el nombre del escenario, la acción del usuario, los milisegundos transcurridos y el número de filas; regístralo en el host builder y envuelve con él tres escenarios: la actualización del dashboard, la búsqueda de tickets y la expansión del árbol de clientes. Ejecuta cada escenario una vez para calentar, luego registra la línea base y escribe un presupuesto de rendimiento que indique el entorno, la configuración de compilación, el tamaño del conjunto de datos y un umbral de aceptación por escenario.

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

- ✓ Solución WisejPerfLab con dashboard, grid de tickets y árbol de clientes sobre datos de prueba
- ✓ Servicio ScenarioProbe que escribe logs PERF estructurados de inicio y fin con milisegundos transcurridos y número de filas
- ✓ Tres escenarios instrumentados: actualización del dashboard, búsqueda de tickets y expansión del árbol de clientes
- ✓ Nota de línea base que registra entorno, configuración de compilación, tamaño del conjunto de datos y procedimiento de calentamiento
- ✓ Presupuesto de rendimiento con un umbral de aceptación por escenario y la herramienta que demostraría cada uno

Por qué tiene esta forma

Un equipo de soporte informa de que WisejPerfLab «se ha vuelto lenta». Esa frase no contiene ninguna medición, ningún escenario y ninguna capa. El dashboard podría estar esperando a SQL, el controlador de actualización podría estar formateando cincuenta mil celdas, el navegador podría estar repintando un grid

que nunca debió renderizarse entero, o el servidor podría estar reteniendo simplemente los formularios de doscientas sesiones que se cerraron hace una hora. Cada una de esas causas tiene una corrección distinta y una herramienta de Visual Studio distinta, y un desarrollador que abre el profiler antes de decidir cuál es recogerá una traza que no puede responder a la pregunta. Este módulo construye primero el modelo mental: qué ocurre realmente entre un clic en el navegador y una pantalla repintada, cuánto cuesta cada uno de esos pasos, y cómo dejar por escrito un escenario y un presupuesto para que los seis módulos siguientes tengan algo con lo que medir.

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
interfaz en el servidor	Los controles, sus propiedades y su estado residen en el proceso del servidor y el navegador renderiza lo que el servidor describe, así que cada objeto de interfaz que creas es memoria del servidor y cada modificación de la interfaz es CPU del servidor más una carga útil.
viaje de ida y vuelta de un evento	El recorrido completo de una acción del usuario: evento del navegador, transporte al servidor, tu controlador, acceso a datos, modificaciones de controles, la actualización serializada de vuelta y el repintado del navegador; es la unidad que el usuario percibe de verdad.
la sesión como unidad de capacidad	Una sesión posee páginas, formularios, controles, temporizadores, selecciones, enlaces de datos y datos en caché; las preguntas de capacidad se responden por sesión, no por petición, porque esa memoria permanece residente entre clics.
push por WebSocket	El canal por el que el servidor envía actualizaciones que el usuario no pidió; abarata la entrega de datos en vivo y encarece una tarea en segundo plano sin limitación, porque cada estado intermedio se convierte en una trama.
los cinco grupos de coste	CPU en tu código, bloqueos y esperas, asignación y retención, acceso a datos y E/S, y carga útil de actualización: la clasificación que decide qué herramienta del profiler puede siquiera ver tu problema.
latencia frente a throughput	La latencia es cuánto espera un usuario; el throughput es cuántas sesiones sostiene el servidor. Una corrección que elimina una espera mejora la latencia; una que elimina trabajo o memoria mejora ambas, y algunas correcciones de latencia cuestan throughput.
escenario y línea base	Una secuencia de acciones de usuario con nombre y repetible, más el entorno, la configuración de compilación, el tamaño del conjunto de datos y las cifras medidas que produjo; sin ella, una traza es ruido y una comparación posterior no tiene sentido.

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 OpsMonitor 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 – «El modelo de rendimiento de Wisej.NET» – del curso de LearnWisej «Rendimiento y profiling», 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/performance-and-profiling/specs/module-01.md>

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

1. Proyecto «OpsMonitor», 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 WisejPerfLab con un dashboard de soporte, un grid de tickets y un árbol de clientes alimentados con datos de prueba, y hazla medible. Añade un servicio ScenarioProbe que envuelva un escenario en un ámbito Stopwatch y escriba logs PERF estructurados de inicio y fin con el nombre del escenario, la acción del usuario, los milisegundos transcurridos y el número de filas; regístralo en el host builder y envuelve con él tres escenarios: la actualización del dashboard, la búsqueda de tickets y la expansión del árbol de clientes. Ejecuta cada escenario una vez para calentar, luego registra la línea base y escribe un presupuesto de rendimiento que indique el entorno, la configuración de compilación, el tamaño del conjunto de datos y un umbral de aceptación por escenario.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Solución WisejPerfLab con dashboard, grid de tickets y árbol de clientes sobre datos de prueba
 - Servicio ScenarioProbe que escribe logs PERF estructurados de inicio y fin con milisegundos transcurridos y número de filas
 - Tres escenarios instrumentados: actualización del dashboard, búsqueda de tickets y expansión del árbol de clientes
 - Nota de línea base que registra entorno, configuración de compilación, tamaño del conjunto de datos y procedimiento de calentamiento
 - Presupuesto de rendimiento con un umbral de aceptación por escenario y la herramienta que demostraría cada uno
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:

- La actualización de tu dashboard registra 1900 ms, pero la CPU del servidor apenas se mueve durante el escenario. ¿Qué grupo de coste descarta eso, a cuál apunta y cuál es la

siguiente medición más barata?

- La sonda mide el controlador. Nombra dos costes que el usuario sufre en el mismo clic y que tu cifra elapsedMs no incluye, y explica cómo medirías cada uno.
- ¿Por qué la nota de línea base tiene que registrar el tamaño del conjunto de datos y la configuración de compilación, si el nombre del escenario y el tiempo transcurrido parecen la parte interesante?

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:

- ✓ La actualización de tu dashboard registra 1900 ms, pero la CPU del servidor apenas se mueve durante el escenario. ¿Qué grupo de coste descarta eso, a cuál apunta y cuál es la siguiente medición más barata?
- ✓ La sonda mide el controlador. Nombra dos costes que el usuario sufre en el mismo clic y que tu cifra elapsedMs no incluye, y explica cómo medirías cada uno.
- ✓ ¿Por qué la nota de línea base tiene que registrar el tamaño del conjunto de datos y la configuración de compilación, si el nombre del escenario y el tiempo transcurrido parecen la parte interesante?

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.



Este ya lo construimos — compara tu versión con la nuestra

La implementación de referencia del Módulo 1 está en el repositorio público de ejemplos. Construye primero la tuya y después abre la nuestra: lo interesante son las diferencias.



[Abrir el ejemplo del Módulo 1 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Performance and Profiling Course/Module 1`.



Nuestro ejemplo es un punto de partida, no un producto terminado

Se construyó igual que estás construyendo el tuyo — con un prompt y luego corregido — y se detiene a propósito donde se detiene el módulo. Espera soluciones provisionales donde iría código de producción, un manejo de errores mínimo y ninguna prueba. Tómallo como una invitación: usa lo que aprendiste en este módulo, junto con el asistente, para llevarlo más lejos. Mejorar código que no escribiste tú es precisamente la habilidad que enseña este curso.

Tres prompts para ir más allá

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

- ✓ **Añade el segundo caso.** *"Amplía este proyecto OpsMonitor 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."*
- ✓ **Demuestra que de verdad funciona.** *"Recorre este proyecto entregable por entregable frente a la especificación del módulo en <https://learnwisej.com/assets/courses/performance-and-profiling/specs/module-01.md> y muéstrame la prueba de que cada uno está realmente terminado — la ruta del código, dónde hacer clic y qué debería ver. Si un entregable solo se cumple en parte, dilo claramente en lugar de justificarlo. Después añade lo que falte."*
- ✓ **Revísalo como un pull request.** *"Explícame este código de OpsMonitor 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."*



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.