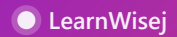


Ejercicio de programación con IA

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

Perfila los dos escenarios dominados por esperas en WisejPerfLab. Ejecuta la búsqueda de tickets con la herramienta Database y asocia cada consulta a la acción de interfaz: la carga del grid lanza una consulta de cliente por cada fila mostrada y la consulta de detalles trae grandes columnas de texto que nadie muestra. Sustituye ambas por una única consulta de proyección sin seguimiento que seleccione solo las columnas del grid y pague el resultado. Después ejecuta la exportación CSV con File I/O y .NET Async, encuentra la llamada bloqueante .Result en el controlador del clic y las escrituras pequeñas repetidas, y traslada la generación a un flujo en segundo plano con Application.StartTask que escriba el archivo en streaming e informe del progreso de forma limitada en lugar de hacer push en cada paso.

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

- ✓ Traza de Database que asocia las consultas a las acciones de interfaz, con el número de consultas N+1 antes de la corrección
- ✓ Única consulta de proyección sin seguimiento con paginación que sustituye las consultas por fila y las que traen datos de más
- ✓ Evidencia de File I/O y .NET Async para la exportación, que nombra la llamada bloqueante y el patrón de escritura
- ✓ Exportación trasladada a una tarea en segundo plano que escribe la salida en streaming y hace push del progreso solo en los pasos significativos
- ✓ Número de consultas, duración de las consultas y tiempo de reloj de la exportación antes/después, con los riesgos restantes

Por qué tiene esta forma

El Módulo 2 midió 1840 ms de tiempo transcurrido y solo encontró 300 ms de CPU. El Módulo 3 eliminó la mayor parte de esa CPU y la pantalla sigue sin ser rápida, porque los 600 ms que faltan nunca fueron cálculo: eran espera. Esta es la categoría peor diagnosticada en una aplicación de gestión: el servidor está inactivo, el gráfico de CPU está plano y el usuario mira un indicador de carga. En algún sitio hay una consulta que se ejecuta una vez por fila, un formulario de detalle que trae una columna que nadie muestra, o un controlador de clic que se bloquea en `Result` mientras un hilo del thread pool no hace nada. Las herramientas que ven esto son Database, File I/O, .NET Async e Instrumentation, y en este módulo la latencia que le queda a WisejPerfLab por fin tiene nombre.

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
herramienta Database	Registra las consultas de ADO.NET y EF Core que lanza un escenario, con su duración y su orden, para que cada consulta pueda asociarse a la acción de interfaz que la provocó.
consulta N+1	Una consulta para listar y luego una más por cada elemento devuelto; la lista va bien con diez filas y se hunde con mil, y es invisible para el profiling de CPU.
traer datos de más	Seleccionar columnas o filas que la pantalla nunca muestra; las grandes columnas de texto y binarias son las culpables habituales, y cuestan tiempo de consulta, transferencia y memoria.
consulta sin seguimiento	Una consulta de solo lectura que no prepara el seguimiento de cambios, adecuada para grids y listas que muestran datos en lugar de editarlos.
herramienta.NET Async	Muestra las actividades asíncronas y dónde esperan, y así se hace visible una cadena secuencial por accidente o bloqueada.
bloquear sobre código async	Llamar a <code>Result</code> o <code>Wait()</code> desde un controlador; retiene un hilo mientras espera y, bajo carga, produce síntomas que parecen agotamiento del thread pool.
exportación en segundo plano	La generación larga de archivos o informes sacada de la ruta inmediata del clic con <code>Application.StartTask</code> , informando del progreso a intervalos significativos y no en cada paso.

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 6. Puedes abrir tú mismo [module-06.md](#) para ver exactamente lo que leerá.



Usar ChatGPT



Usar Claude

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.

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 6 - «Base de datos, E/S de archivos, async y esperas externas» - 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-06.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: Perfila los dos escenarios dominados por esperas en `WisejPerfLab`. Ejecuta la búsqueda de tickets con la herramienta Database y asocia cada consulta a la acción de interfaz: la carga del grid lanza una consulta de cliente por cada fila mostrada y la consulta de detalles trae grandes columnas de texto que nadie muestra. Sustituye ambas por una única consulta de proyección sin seguimiento que seleccione solo las columnas del grid y pague el resultado. Después ejecuta la exportación CSV con File I/O y .NET Async, encuentra la llamada bloqueante `.Result` en el controlador del clic y las escrituras pequeñas repetidas, y traslada la generación a un flujo en segundo plano con `Application.StartTask` que escriba el archivo en streaming e informe del progreso de forma limitada en lugar de hacer push en cada paso.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Traza de Database que asocia las consultas a las acciones de interfaz, con el número de consultas N+1 antes de la corrección
 - Única consulta de proyección sin seguimiento con paginación que sustituye las consultas por fila y las que traen datos de más
 - Evidencia de File I/O y .NET Async para la exportación, que nombra la llamada bloqueante y el patrón de escritura
 - Exportación trasladada a una tarea en segundo plano que escribe la salida en streaming y hace push del progreso solo en los pasos significativos
 - Número de consultas, duración de las consultas y tiempo de reloj de la exportación antes/después, con los riesgos restantes
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 búsqueda lanza ahora una consulta en lugar de doscientas una, pero el tiempo transcurrido solo se ha reducido a la mitad. ¿Dónde está el resto del tiempo y qué

herramienta usarías para averiguarlo?

- Un compañero dice que la exportación está bien porque.Result «solo bloquea un hilo». Explica qué ocurre con cincuenta exportaciones concurrentes y qué pantallas notarías el usuario que se ralentizan.
- De lo que este laboratorio guarda en caché, ¿qué es seguro compartir entre sesiones y qué no, y qué regla usaste para decidirlo?

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 búsqueda lanza ahora una consulta en lugar de doscientas una, pero el tiempo transcurrido solo se ha reducido a la mitad. ¿Dónde está el resto del tiempo y qué herramienta usarías para averiguarlo?
- ✓ Un compañero dice que la exportación está bien porque.Result «solo bloquea un hilo». Explica qué ocurre con cincuenta exportaciones concurrentes y qué pantallas notarías el usuario que se ralentizan.
- ✓ De lo que este laboratorio guarda en caché, ¿qué es seguro compartir entre sesiones y qué no, y qué regla usaste para decidirlo?

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 6 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 6 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Performance and Profiling Course/Module 6`.



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-06.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.