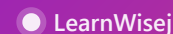


Ejercicio de programación con IA

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.



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

Todas las demás secciones de este módulo explican cómo funciona «Controles de datos, cargas útiles del navegador y grandes superficies de interfaz». 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

Reduce las dos superficies más grandes de WisejPerfLab. El grid de tickets enlaza 50.000 entidades completas con propiedades de navegación: sustituye el enlace por una proyección TicketGridRow con solo las columnas visibles, establece VirtualMode y RowCount y sirve las celdas desde un controlador CellValueNeeded respaldado por un servicio de consultas paginadas. El árbol de clientes construye todos los nodos al arrancar: carga los hijos al expandir, usando un marcador de recuento para las ramas contraídas. Mide cada pantalla antes y después con CPU Usage y .NET Object Allocation en el servidor, y con el panel de red del navegador para el tamaño de la carga útil de actualización.

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

- ✓ Proyección TicketGridRow enlazada en lugar de entidades completas, solo con las columnas mostradas
- ✓ Grid en modo virtual con RowCount y un controlador CellValueNeeded servido desde un servicio de consultas paginadas
- ✓ Árbol de clientes que carga los nodos hijos al expandir con un marcador de recuento para las ramas contraídas
- ✓ Cifras de CPU y asignaciones antes/después para los escenarios de carga del grid y expansión del árbol
- ✓ Evidencia de red del navegador que compara el tamaño de la carga útil de actualización antes y después, con la advertencia sobre la compresión

Por qué tiene esta forma

Las grandes superficies de interfaz son la fuente más habitual de problemas de rendimiento en Wisej.NET, y son el grupo de coste que el profiler no puede ver del todo. El grid de tickets de WisejPerfLab enlaza

cincuenta mil entidades con sus propiedades de navegación; el árbol de clientes construye todos los nodos al arrancar. En escritorio ese patrón era aceptable, porque el control y los datos vivían en el mismo proceso que la pantalla. Aquí el mismo código paga cuatro veces: memoria del servidor para retener los objetos, CPU del servidor para construir y serializar el estado de la interfaz, red para transportar la actualización y tiempo del navegador para renderizarla. En este módulo la mejor optimización no suele ser un algoritmo más rápido, sino que exista menos interfaz a la vez.

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
fila de proyección	Un tipo pequeño que contiene solo las columnas que muestra una pantalla, enlazado en lugar del grafo de entidades, para que ni el servidor ni el navegador carguen con datos que nadie mira.
modo virtual	El grid guarda un número de filas en lugar de objetos de fila y pide los valores de las celdas a medida que los necesita, lo que elimina los objetos por fila y el renderizado de filas a las que nadie ha llegado.
CellValueNeeded	El evento que proporciona un valor bajo demanda en modo virtual; se ejecuta muy a menudo, así que es una ruta crítica por construcción y debe devolver el valor rápido.
paginación y filtrado en el servidor	Pedir a la base de datos la página que el usuario está mirando, en lugar de cargarlo todo y filtrar en el código de interfaz.
carga diferida de nodos	Construir los hijos del árbol cuando se expande una rama, con un marcador de recuento mientras está contraída, en lugar de materializar toda la jerarquía al arrancar.
carga útil de actualización	La diferencia serializada que el servidor envía tras un controlador; las cargas grandes o frecuentes cuestan tiempo de red y de navegador que ninguna herramienta de Visual Studio muestra.
la compresión no ahorra el trabajo	Wisej.NET comprime el tráfico HTTP y WebSocket, lo que reduce los bytes pero no abarata producir, transmitir o renderizar una actualización innecesaria.

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



Usar Claude

Abre [claude.ai](#) con el mismo prompt ya cargado. Si usas Claude Code, puedes

Codex; cualquier plan que pueda navegar por la web descargará la especificación.

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 5 – «Controles de datos, cargas útiles del navegador y grandes superficies de interfaz» – 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-05.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: Reduce las dos superficies más grandes de WisejPerfLab. El grid de tickets enlaza 50.000 entidades completas con propiedades de navegación: sustituye el enlace por una proyección `TicketGridRow` con solo las columnas visibles, establece `VirtualMode` y `RowCount` y sirve las celdas desde un controlador `CellValueNeeded` respaldado por un servicio de consultas paginadas. El árbol de clientes construye todos los nodos al arrancar: carga los hijos al expandir, usando un marcador de recuento para las ramas contraídas. Mide cada pantalla antes y después con CPU Usage y .NET Object Allocation en el servidor, y con el panel de red del navegador para el tamaño de la carga útil de actualización.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Proyección `TicketGridRow` enlazada en lugar de entidades completas, solo con las columnas mostradas
 - Grid en modo virtual con `RowCount` y un controlador `CellValueNeeded` servido desde un servicio de consultas paginadas
 - Árbol de clientes que carga los nodos hijos al expandir con un marcador de recuento para las ramas contraídas
 - Cifras de CPU y asignaciones antes/después para los escenarios de carga del grid y expansión del árbol
 - Evidencia de red del navegador que compara el tamaño de la carga útil de actualización antes y después, con la advertencia sobre la compresión
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 modo virtual eliminó los objetos de fila, pero tu traza de CPU apenas mejoró. ¿Qué te dice eso sobre dónde estaba realmente el coste y qué medición mirarías a continuación?
- Tu controlador `CellValueNeeded` es correcto pero lento con un desplazamiento rápido.

Enumera tres cosas que podría estar haciendo y que un controlador por celda nunca debe hacer.

– La carga útil comprimida de la expansión del árbol apenas cambió tras la carga diferida, pero los usuarios dicen que la pantalla parece más rápida. Da una explicación plausible que el recuento de bytes por sí solo ocultaría.

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 modo virtual eliminó los objetos de fila, pero tu traza de CPU apenas mejoró. ¿Qué te dice eso sobre dónde estaba realmente el coste y qué medición mirarías a continuación?
- ✓ Tu controlador CellValueNeeded es correcto pero lento con un desplazamiento rápido. Enumera tres cosas que podría estar haciendo y que un controlador por celda nunca debe hacer.
- ✓ La carga útil comprimida de la expansión del árbol apenas cambió tras la carga diferida, pero los usuarios dicen que la pantalla parece más rápida. Da una explicación plausible que el recuento de bytes por sí solo ocultaría.

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 5 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 5 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Performance and Profiling Course/Module 5`.



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.

- ✓ **Haz que aguante un mal día.** *"Revisa este proyecto OpsMonitor 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 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-05.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."*



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.