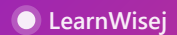


Ejercicio de programación con IA

Construye el ejemplo VisualOperationsStudio 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.



Instrúyelo, léelo y luego rómpelo.

Este ejercicio tiene tres partes, y solo la primera trata del prompt. Le das al asistente la especificación del módulo y recibes una solución VisualOperationsStudio; lees lo que escribió y decides si de verdad hace lo que pedía el módulo; y después lo llevas más allá de lo que pedía el módulo. En la segunda y la tercera parte es donde se aprende.

Qué estás pidiendo

Dota a VisualOperationsStudio de una cuadrícula de operaciones con dos columnas dibujadas por el usuario. Añade columnas explícitas a gridOperations, activa UserPaint = true solo en las columnas Health y Trend y gestiona CellPaint una única vez: sal de inmediato cuando e.RowIndex sea negativo o e.ColumnIndex no sea una de las dos columnas objetivo, y después dibuja una barra de progreso coloreada por umbrales para Health y una sparkline de polilínea para Trend dentro de e.ClipRectangle. Lee los valores de cada fila del objeto MachineStatus enlazado en lugar de buscar en una lista dentro del controlador, respeta los colores de la fila seleccionada y mantén el controlador libre de efectos secundarios. Enlaza 1000 filas generadas, desplázate por ellas y después construye el mismo indicador de estado una segunda vez como celda AllowHtml, para poder dejar por escrito cuál de los dos llevarías a producción.

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

- ✓ gridOperations con columnas explícitas y UserPaint = true solo en las columnas Health y Trend
- ✓ Un único controlador de CellPaint que comprueba e.RowIndex y e.ColumnIndex antes de dibujar nada
- ✓ Celda de progreso coloreada por umbrales y celda de sparkline de polilínea dibujadas dentro de e.ClipRectangle
- ✓ Valores de fila leídos del objeto MachineStatus enlazado, sin búsquedas, reservas de memoria ni creación de controles por celda
- ✓ Una columna de estado con AllowHtml construida en paralelo, más una nota que compare ambas en un desplazamiento por 1000 filas

Por qué tiene esta forma

El Módulo 2 dio a VisualOperationsStudio un TelemetryGauge: un control, una máquina, unos cientos de redibujados por hora. La cuadrícula de operaciones es otra cosa. Un supervisor de turno abre una lista de

mil máquinas y quiere ver de un vistazo cuáles están cerca de su límite y cuáles llevan toda la mañana desviándose. Unos números en una columna no consiguen eso, y un indicador por fila serían mil controles del lado del servidor en una sola sesión. Es la propia cuadrícula la que tiene que dibujar, una vez por celda visible, y parar en cuanto la celda se desplaza fuera de la vista.

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
UserPaint	El interruptor explícito: <code>DataGridViewCell.UserPaint = true</code> para una celda, <code>DataGridViewColumn.UserPaint = true</code> para una columna entera. Sin él, la cuadrícula nunca te pide que dibujes.
CellPaint	El evento a nivel de cuadrícula que se dispara para las celdas dibujadas por el usuario cuando hay que dibujarlas; te suscribes una vez en la cuadrícula, no una vez por fila, y distribuyes dentro del controlador.
<code>DataGridViewCellPaintEventArgs</code>	Deriva de <code>PaintEventArgs</code> , así que lleva <code>Graphics</code> y <code>ClipRectangle</code> , y añade <code>RowIndex</code> y <code>ColumnIndex</code> para que un solo controlador pueda atender varias columnas.
sin ayudantes de WinForms	Los argumentos de evento documentados de Wisej.NET no exponen los métodos de conveniencia de WinForms <code>PaintBackground()</code> y <code>PaintContent()</code> . Tu controlador es dueño de todo el resultado visual de esa celda.
<code>e.ClipRectangle</code> como superficie	El rectángulo de destino local de la celda. Las columnas se pueden redimensionar y la cuadrícula puede disponerse de forma adaptable, así que cada coordenada que dibujes tiene que derivarse de él y no de una constante.
ayudantes de dibujo puros	Métodos con la forma <code>DrawProgressCell(Graphics g, Rectangle rect, double value)</code> , para que el controlador se convierta en código de distribución y ese mismo ayudante pueda dibujar más adelante en un mapa de bits fuera de pantalla.
contenido de celda <code>AllowHtml</code>	La alternativa nativa del navegador: iconos, spans, etiquetas y CSS representados por el cliente, y la razón por la que la documentación reserva <code>CellPaint</code> para elementos visuales demasiado complejos para HTML.

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 3. Puedes abrir tú mismo [module-03.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 VisualOperationsStudio 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 3 - «Dibujo de celdas en DataGridView» - del curso de LearnWisej «Dibujo y pintura en Wisej.NET», 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/drawing-and-painting/specs/module-03.md>

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

1. Proyecto «VisualOperationsStudio», 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: Dota a `VisualOperationsStudio` de una cuadrícula de operaciones con dos columnas dibujadas por el usuario. Añade columnas explícitas a `gridOperations`, activa `UserPaint = true` solo en las columnas `Health` y `Trend` y gestiona `CellPaint` una única vez: sal de inmediato cuando `e.RowIndex` sea negativo o `e.ColumnIndex` no sea una de las dos columnas objetivo, y después dibuja una barra de progreso coloreada por umbrales para `Health` y una sparkline de polilínea para `Trend` dentro de `e.ClipRectangle`. Lee los valores de cada fila del objeto `MachineStatus` enlazado en lugar de buscar en una lista dentro del controlador, respeta los colores de la fila seleccionada y mantén el controlador libre de efectos secundarios. Enlaza 1000 filas generadas, desplázate por ellas y después construye el mismo indicador de estado una segunda vez como celda `AllowHtml`, para poder dejar por escrito cuál de los dos llevarías a producción.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `gridOperations` con columnas explícitas y `UserPaint = true` solo en las columnas `Health` y `Trend`
 - Un único controlador de `CellPaint` que comprueba `e.RowIndex` y `e.ColumnIndex` antes de dibujar nada
 - Celda de progreso coloreada por umbrales y celda de sparkline de polilínea dibujadas dentro de `e.ClipRectangle`
 - Valores de fila leídos del objeto `MachineStatus` enlazado, sin búsquedas, reservas de memoria ni creación de controles por celda
 - Una columna de estado con `AllowHtml` construida en paralelo, más una nota que compare ambas en un desplazamiento por 1000 filas
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 compañero pega un ejemplo de dibujo de celdas de WinForms que empieza llamando a

`e.PaintBackground(...)`. Explica qué te dan en su lugar los argumentos de evento de Wisej.NET y qué tiene que decidir ahora el controlador por sí mismo.

- Tu ayudante de sparkline fija en el código un ancho de trazado de 100 píxeles y se ve bien en tu pantalla. Describe dos acciones del usuario que lo rompen y qué debería leer el ayudante en su lugar.
- La columna de estado en HTML y la columna Health dibujada muestran las dos una máquina en estado crítico. Argumenta cuál llevarías a producción en una cuadrícula que el personal de soporte busca y lee con un lector de pantalla, y qué perderías en cada caso.

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 compañero pega un ejemplo de dibujo de celdas de WinForms que empieza llamando a `e.PaintBackground(...)`. Explica qué te dan en su lugar los argumentos de evento de Wisej.NET y qué tiene que decidir ahora el controlador por sí mismo.
- ✓ Tu ayudante de sparkline fija en el código un ancho de trazado de 100 píxeles y se ve bien en tu pantalla. Describe dos acciones del usuario que lo rompen y qué debería leer el ayudante en su lugar.
- ✓ La columna de estado en HTML y la columna Health dibujada muestran las dos una máquina en estado crítico. Argumenta cuál llevarías a producción en una cuadrícula que el personal de soporte busca y lee con un lector de pantalla, y qué perderías en cada caso.

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 VisualOperationsStudio 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 VisualOperationsStudio 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 VisualOperationsStudio 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.