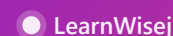


Ejercicio de programación con IA

Construye el ejemplo OperationsConsole 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.



Escribir el código ya es la mitad fácil.

Conseguir que un asistente moderno genere código que funcione para «Listas, árboles, repetidores y datos jerárquicos» solo requiere un buen prompt. Juzgarlo, corregirlo y ampliarlo requiere todo lo que ha cubierto este módulo. Ese es el reparto sobre el que se construye este ejercicio: tú aportas la especificación y la revisión; el asistente, el tecleo.

Qué estás pidiendo

Construye el explorador de documentos de Operations Console: un TreeView de categorías o carpetas con un ID estable asociado a cada nodo, un ListView de documentos vinculado al nodo seleccionado, carga diferida o modo virtual en al menos una ruta de lista o árbol, iconos o claves de imagen para los distintos tipos de elemento, y un UserControl de detalle que se carga a través de una capa de servicio o view-model cada vez que cambia la selección.

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

- ✓ TreeView de categorías con un ID estable guardado en cada nodo
- ✓ ListView de documentos vinculado al nodo seleccionado del árbol
- ✓ Carga diferida o modo virtual en al menos una ruta de lista o árbol
- ✓ UserControl de detalle cargado a través de una capa de servicio o view-model al seleccionar
- ✓ Iconos o claves de imagen para los distintos tipos de elemento

Por qué tiene esta forma

Los controles de lista y de árbol suelen tratarse como simples widgets de selección, pero en una aplicación de gestión son el mapa del usuario: carpetas, clientes, pedidos, procedimientos, roles, tareas, documentos, catálogos de productos y los propios módulos de la aplicación. El explorador de documentos de Operations Console que construyes en este módulo tiene que seguir siendo rápido cuando el árbol de categorías es profundo y la lista de documentos es larga, y cada clic tiene que corresponder a un registro de negocio real y no a un fragmento de texto mostrado.

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
forma de los datos	La primera pregunta de diseño: ¿los datos son planos, agrupados, jerárquicos, repetidos como tarjetas o un conjunto de propiedades? La respuesta determina el control.
vistas de ListView	Un único control que muestra la misma colección como LargeIcon, SmallIcon, Details o Tile, la elección natural para pantallas tipo explorador de archivos o catálogo.
modo virtual	Un modo de ListView en el que la aplicación establece VirtualListSize y responde a RetrieveVirtualItem, de modo que los elementos solo existen para las filas que pide el control.
carga diferida	Cargar solo los niveles superiores de un árbol al arrancar y obtener los hijos cuando se expande un nodo, para que el arranque siga siendo barato y la interfaz ágil.
desplazamiento virtual y precarga	Renderizar en el navegador solo las filas visibles de una lista o un árbol grandes, con PrefetchItems prerrenderizando algunas filas extra para un desplazamiento fluido.
identidad estable	Un ID de negocio único guardado en el Tag o el Name del nodo o elemento, o en un mapa del view model, nunca derivado del texto mostrado.
la selección como evento	Un cambio de selección obtiene un ID y pide los datos a un servicio; nunca se ejecuta una consulta dentro del código que construye nodos o elementos.

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 4. Puedes abrir tú mismo [module-04.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 OperationsConsole 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 4 - «Listas, árboles, repetidores y datos jerárquicos» - del curso de LearnWisej «Dominar la biblioteca de controles», 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/mastering-the-control-library/specs/module-04.md>

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

1. Proyecto «OperationsConsole», 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: Construye el explorador de documentos de Operations Console: un TreeView de categorías o carpetas con un ID estable asociado a cada nodo, un ListView de documentos vinculado al nodo seleccionado, carga diferida o modo virtual en al menos una ruta de lista o árbol, iconos o claves de imagen para los distintos tipos de elemento, y un UserControl de detalle que se carga a través de una capa de servicio o view-model cada vez que cambia la selección.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - TreeView de categorías con un ID estable guardado en cada nodo
 - ListView de documentos vinculado al nodo seleccionado del árbol
 - Carga diferida o modo virtual en al menos una ruta de lista o árbol
 - UserControl de detalle cargado a través de una capa de servicio o view-model al seleccionar
 - Iconos o claves de imagen para los distintos tipos de elemento
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:

- Dos categorías de clientes distintos se llaman ambas «Contracts». ¿Qué usa exactamente tu manejador de selección para distinguirlas y dónde se guarda ese valor?
- Si el número de documentos de una categoría pasara de 300 a 300.000, ¿qué parte de tu explorador fallaría primero y qué propiedad o evento cambiarías para solucionarlo?
- ¿Qué líneas de tu manejador de selección podrían pasar a una prueba unitaria sin navegador, y qué líneas corresponden en cambio al UserControl de detalle?

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:

- ✓ Dos categorías de clientes distintos se llaman ambas «Contracts». ¿Qué usa exactamente tu manejador de selección para distinguirlas y dónde se guarda ese valor?
- ✓ Si el número de documentos de una categoría pasara de 300 a 300.000, ¿qué parte de tu explorador fallaría primero y qué propiedad o evento cambiarías para solucionarlo?
- ✓ ¿Qué líneas de tu manejador de selección podrían pasar a una prueba unitaria sin navegador, y qué líneas corresponden en cambio al UserControl de detalle?

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 4 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 4 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Mastering the Control Library Course/Module 4`.



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.

- ✓ **Revísalo como un pull request.** *"Explicame este código de OperationsConsole 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 OperationsConsole 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 OperationsConsole 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.