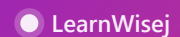


Ejercicio de programación con IA

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.



El asistente pone las manos rápidas. El arquitecto sigues siendo tú.

Un asistente puede generar el ejemplo de este módulo en minutos. Lo que no puede hacer es decidir si el resultado es el diseño adecuado para tu aplicación — esa decisión sigue siendo tuya. Tú pones el criterio: dale unas buenas instrucciones y luego contrasta lo que te devuelva con lo que te acaba de enseñar el Módulo 7.

Qué estás pidiendo

Envuelve un pequeño widget JavaScript de valoración o indicador con el control Widget, o configura un paquete de extensión de Wisej.NET: carga sus packages JavaScript y CSS, lanza un evento del lado del servidor desde el cliente con el patrón de eventos de widget, llama a una función del lado del cliente desde el servidor con CallAsync o Eval, aplícale un tema acorde con la consola y, por último, pule el proyecto final completo de Operations Console y prepara una demo de 5 minutos.

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

- ✓ Pantalla basada en Widget o en una extensión con sus packages JavaScript y CSS cargados
- ✓ Evento de cliente a servidor lanzado con el patrón de eventos de widget o un evento nativo de la extensión
- ✓ Llamada de servidor a cliente mediante CallAsync, Eval o la API de la extensión
- ✓ Contrato cliente/servidor documentado: packages, propiedades, eventos y formato del payload
- ✓ Proyecto final de Operations Console pulido con un guion de demo de 5 minutos

Por qué tiene esta forma

Tarde o temprano un producto necesita algo que la biblioteca nativa no incluye: una capacidad especializada del navegador, una librería JavaScript de terceros, una visualización personalizada o una integración reutilizable. La tentación es lanzarse a JavaScript y perder el modelo de Wisej.NET: el estado en el servidor, los temas, el diseñador y un flujo de eventos que otro desarrollador pueda seguir. Este módulo integra un widget de cliente sin renunciar a nada de eso y después completa el proyecto final de Operations Console.

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
control nativo	Un control que implementa tanto el lado del servidor como su widget del navegador, así que ya tiene soporte en tiempo de diseño, temas, eventos y estado en el servidor; siempre la primera opción.
tipos de extensión	Control, Widget, Component, Extender Provider e Icon Pack: nombrar el tipo evita que «reutilizable» se confunda con «nuevo control nativo».
control Widget	Un contenedor de cliente de uso general que carga packages JavaScript y CSS, ejecuta un script de inicialización e intercambia eventos y datos con el código del servidor.
Packages e InitScript	La lista de packages declara los archivos de cliente de los que depende el widget; el InitScript crea el objeto JavaScript, conecta sus eventos de cliente y expone solo las funciones que necesita el servidor.
WidgetEvent	El evento del servidor que se lanza cuando el código de cliente llama a fireWidgetEvent; su tipo y su payload de datos llegan desde el navegador y deben validarse como cualquier otra entrada del usuario.
CallAsync / Eval	La mitad de la pasarela que va del servidor al cliente: llamar a una función expuesta por el widget, o evaluar un script, sin salir de C#.
contrato del widget	Las propiedades, los eventos, el formato del payload, las dependencias de packages y las reglas de temas documentados que permiten a otro desarrollador, o a un asistente de IA con el contexto adecuado, mantener el widget.

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 7. Puedes abrir tú mismo [module-07.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 7 – «Widgets personalizados, extensiones, temas y proyecto final» – 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-07.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: Envuelve un pequeño widget JavaScript de valoración o indicador con el control Widget, o configura un paquete de extensión de Wisej.NET: carga sus packages JavaScript y CSS, lanza un evento del lado del servidor desde el cliente con el patrón de eventos de widget, llama a una función del lado del cliente desde el servidor con `CallAsync` o `Eval`, aplícale un tema acorde con la consola y, por último, pule el proyecto final completo de Operations Console y prepara una demo de 5 minutos.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Pantalla basada en Widget o en una extensión con sus packages JavaScript y CSS cargados
 - Evento de cliente a servidor lanzado con el patrón de eventos de widget o un evento nativo de la extensión
 - Llamada de servidor a cliente mediante `CallAsync`, `Eval` o la API de la extensión
 - Contrato cliente/servidor documentado: packages, propiedades, eventos y formato del payload
 - Proyecto final de Operations Console pulido con un guion de demo de 5 minutos
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:

- ¿Qué pasaría si un usuario modificara el payload de `ratingChanged` en las herramientas de desarrollo del navegador antes de que llegue a tu manejador, y qué línea de tu código lo impide?
- Tu widget aparece en blanco después de ocultar y volver a mostrar la pestaña Widgets: ¿qué tres cosas compruebas, en qué orden, y por qué el HTML de arranque no es la solución?
- ¿Qué regla de tu widget podrías trasladar a JavaScript como optimización y cuál no debe salir nunca del servidor?

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:

- ✓ ¿Qué pasaría si un usuario modificara el payload de `ratingChanged` en las herramientas de desarrollo del navegador antes de que llegue a tu manejador, y qué línea de tu código lo impide?
- ✓ Tu widget aparece en blanco después de ocultar y volver a mostrar la pestaña Widgets: ¿qué tres cosas compruebas, en qué orden, y por qué el HTML de arranque no es la solución?
- ✓ ¿Qué regla de tu widget podrías trasladar a JavaScript como optimización y cuál no debe salir nunca del servidor?

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 7 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 7 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Mastering the Control Library Course/Module 7`.



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 OperationsConsole 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/mastering-the-control-library/specs/module-07.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 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."*



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.