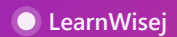


Ejercicio de programación con IA

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 2.

Qué estás pidiendo

Construye un UserControl CustomerEditor para Operations Console con campos etiquetados para nombre, email, estado, fecha de inicio, límite de crédito y tipo de cliente, usando el editor adecuado para cada valor en lugar de un TextBox para todo. Añade mensajes de ErrorProvider para campos obligatorios, formato del email, rango del límite de crédito y fechas no válidas; comandos Save, Reset y Validate con comportamiento deshabilitado y ocupado durante el guardado; y un Toast o MessageBox cuando termine la validación o el guardado.

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

- ✓ UserControl CustomerEditor con campos etiquetados para nombre, email, estado, fecha de inicio, límite de crédito y tipo de cliente
- ✓ Editores adecuados: TextBox, DateTimePicker, NumericUpDown y ComboBox en lugar de texto libre en todas partes
- ✓ Mensajes de ErrorProvider para campos obligatorios, formato del email, rango del límite de crédito y fechas no válidas
- ✓ Comandos Save, Reset y Validate con comportamiento deshabilitado y ocupado durante el guardado
- ✓ Feedback con Toast o MessageBox cuando termine la validación o el guardado

Por qué tiene esta forma

Los editores son el primer lugar donde una biblioteca de controles se convierte en un sistema de negocio. Un formulario que parece correcto pero acepta entradas ambiguas o no válidas genera tickets de soporte y scripts de limpieza mucho después de haberse publicado. Los editores de Wisej.NET siguen el conocido modelo de componentes de .NET pero se ejecutan en el navegador, así que una misma regla de negocio

puede aplicarse en varias capas: propiedades del editor, extenders de validación, manejadores de eventos, enlace de datos, validación del servicio y restricciones de la base de datos. La función de la capa de interfaz es evitar errores evidentes y explicarlos con claridad; nunca debe ser la única capa que protege la aplicación.

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
editor adecuado al valor	El editor cuyo tipo ya restringe la entrada: DateTimePicker para fechas, NumericUpDown para números acotados, ComboBox para valores de una lista, de modo que el servidor nunca analiza texto ambiguo.
modelo de comandos	Una separación deliberada entre comandos principales (Save, Approve, Search, Send, Export), comandos secundarios (ToolBar, MenuBar, SplitButton, menú contextual) y comandos destructivos que necesitan confirmación o una ubicación cuidadosa.
ErrorProvider	El extender que asocia un mensaje concreto y accionable al control que está mal, en lugar de un genérico «Valor no válido» en otro lugar de la página.
feedback bloqueante frente a no bloqueante	MessageBox interrumpe para pedir una decisión o una confirmación crítica; AlertBox y Toast informan del estado sin detener el flujo de trabajo.
validadores con nombre	Métodos como ValidateEmail o ValidateCreditLimit que contienen una regla cada uno, para que la regla se pueda probar y explicar en lugar de quedar enterrada en un manejador de clic.
estado ocupado	Deshabilitar Save y mostrar el progreso mientras se ejecuta una operación, lo que evita envíos duplicados y forma parte de la fiabilidad, no de la decoración.
UserControl de edición	Una composición de etiquetas, editores, un ErrorProvider y un método de validación que devuelve un objeto de resultado, para que el resto de la aplicación nunca sepa cómo se comprueba el campo de email.

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 2. Puedes abrir tú mismo [module-02.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 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 2 - «Editores, botones, validación y feedback» - 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-02.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 un `UserControl CustomerEditor` para `Operations Console` con campos etiquetados para nombre, email, estado, fecha de inicio, límite de crédito y tipo de cliente, usando el editor adecuado para cada valor en lugar de un `TextBox` para todo. Añade mensajes de `ErrorProvider` para campos obligatorios, formato del email, rango del límite de crédito y fechas no válidas; comandos `Save`, `Reset` y `Validate` con comportamiento deshabilitado y ocupado durante el guardado; y un `Toast` o `AlertBox` cuando termine la validación o el guardado.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `UserControl CustomerEditor` con campos etiquetados para nombre, email, estado, fecha de inicio, límite de crédito y tipo de cliente
 - Editores adecuados: `TextBox`, `DateTimePicker`, `NumericUpDown` y `ComboBox` en lugar de texto libre en todas partes
 - Mensajes de `ErrorProvider` para campos obligatorios, formato del email, rango del límite de crédito y fechas no válidas
 - Comandos `Save`, `Reset` y `Validate` con comportamiento deshabilitado y ocupado durante el guardado
 - Feedback con `Toast` o `AlertBox` cuando termine la validación o el guardado
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 errores de validación ya no pueden producirse en absoluto gracias al editor que elegiste, y cuáles siguen necesitando un mensaje de `ErrorProvider`?
- Si el usuario hace clic dos veces en `Save` en un segundo, ¿qué hace tu pantalla y en qué parte del código está garantizado?
- ¿Qué sabe la página del shell sobre cómo se valida el campo de email, y qué tendría que cambiar si la regla se trasladara a un servicio?

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 errores de validación ya no pueden producirse en absoluto gracias al editor que elegiste, y cuáles siguen necesitando un mensaje de ErrorProvider?
- ✓ Si el usuario hace clic dos veces en Save en un segundo, ¿qué hace tu pantalla y en qué parte del código está garantizado?
- ✓ ¿Qué sabe la página del shell sobre cómo se valida el campo de email, y qué tendría que cambiar si la regla se trasladara a un servicio?

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 2 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 2 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Mastering the Control Library Course/Module 2`.



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-02.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.