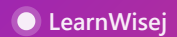


Ejercicio de programación con IA

Construye el ejemplo ValidationClinic 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.



No vas a teclear este ejemplo. Vas a dirigirlo.

Ya casi nadie escribe código Wisej.NET partiendo de un archivo vacío — un asistente escribe el primer borrador y el desarrollador decide si es correcto. Eso solo funciona si dominas el tema, y eso es lo que te dio el resto de este módulo. Aquí le pasas ese conocimiento a ChatGPT o a Claude, recibes el ejemplo de este módulo como una solución de Visual Studio que funciona y lo revisas como revisarías el pull request de un compañero.

Qué estás pidiendo

Haz que la cuadrícula de contactos de ValidationClinic rechace los datos incorrectos. Gestiona `gridContacts.CellValidating`: lee `e.FormattedValue` en lugar del valor de la celda, borra el `ErrorText` de la celda y rechaza un valor de `EmailColumn` vacío o sin `@` y un valor de `AgeColumn` fuera del rango de 0 a 120 con `e.Cancel = true` y un `ErrorText` en la celda. Añade un método `ValidateRow` que establezca el `ErrorText` de la fila y el error de la celda `ClosedDateColumn` cuando `Status` sea `Closed` sin fecha de cierre, borra los errores obsoletos en `CellValidated` y gestiona `DataError` con `ThrowException = false`, un mensaje comprensible en la celda y un registro de depuración de la excepción. Omite la fila nueva de marcador de posición hasta que contenga datos del usuario y bloquea `Save` mientras alguna fila tenga todavía un error.

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

- ✓ Controlador `CellValidating` que lee `e.FormattedValue` y valida las columnas de correo electrónico y edad
- ✓ Método `ValidateRow` que establece el `ErrorText` de la fila y el error de la celda de fecha de cierre para los contactos `Closed`
- ✓ Controlador `CellValidated` que borra los errores de celda obsoletos
- ✓ Controlador `DataError` con `ThrowException = false`, un mensaje comprensible y la excepción registrada
- ✓ Protección de `Save` que omite la fila nueva de marcador de posición y se niega a guardar mientras alguna fila tenga un error

Por qué tiene esta forma

Los módulos 1 a 5 protegieron el formulario de alta de ValidationClinic, un contacto cada vez. La cuadrícula de contactos es distinta: un usuario puede recorrer veinte filas con Tab, escribir un correo electrónico sin @, introducir «abc» como edad y marcar un contacto como Closed sin fecha de cierre, todo sin abrir ningún formulario. Una celda de la cuadrícula no es un TextBox, así que los controladores Validating y el ErrorProvider que ya has escrito no llegan a ella. Este módulo da a la cuadrícula sus propias reglas de campo, reglas de fila y reglas del origen de datos, y se asegura de que ninguna de ellas convierta un error de escritura normal en un bloqueo o en una pantalla atascada.

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
CellValidating	El evento de la cuadrícula que se desencadena cuando la celda actual está a punto de perder el foco; es la versión de la cuadrícula del evento Validating de un control y el lugar para rechazar un valor concreto.
e.FormattedValue	El valor exactamente como el usuario lo ha escrito en el editor, antes de confirmarse en la celda o en el origen de datos; leer en su lugar el Value de la celda te da el valor antiguo.
e.Cancel	Establecerlo en true impide terminar la edición, así que el cursor se queda en la celda incorrecta hasta que el usuario la corrija o cancele la edición.
CellValidated	Se desencadena después de que una celda haya superado la validación; un lugar seguro para borrar un error que ya no se aplica o actualizar un resumen.
ErrorText de celda	Un mensaje en una sola celda, mostrado con un icono de error dentro de esa celda; úsalo para un problema con un único valor.
ErrorText de fila	Un mensaje en toda la fila, mostrado en el encabezado de fila; úsalo para una regla que depende de varias celdas de la misma fila.
DataError	Se desencadena cuando la cuadrícula no puede convertir, dar formato o confirmar un valor; con e.ThrowException = false sustituyes la excepción por un mensaje comprensible.

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 6. Puedes abrir tú mismo [module-06.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 ValidationClinic 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 6 – «Validación de celdas y errores de fila en DataGridView» – del curso de LearnWisej «Validación 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/validation-in-wisej-net/specs/module-06.md>

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

1. Proyecto «ValidationClinic», 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: Haz que la cuadrícula de contactos de ValidationClinic rechace los datos incorrectos. Gestiona `gridContacts.CellValidating`: lee `e.FormattedValue` en lugar del valor de la celda, borra el `ErrorText` de la celda y rechaza un valor de `EmailColumn` vacío o sin @ y un valor de `AgeColumn` fuera del rango de 0 a 120 con `e.Cancel = true` y un `ErrorText` en la celda. Añade un método `ValidateRow` que establezca el `ErrorText` de la fila y el error de la celda `ClosedDateColumn` cuando `Status` sea `Closed` sin fecha de cierre, borra los errores obsoletos en `CellValidated` y gestiona `DataError` con `ThrowException = false`, un mensaje comprensible en la celda y un registro de depuración de la excepción. Omite la fila nueva de marcador de posición hasta que contenga datos del usuario y bloquea `Save` mientras alguna fila tenga todavía un error.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Controlador `CellValidating` que lee `e.FormattedValue` y valida las columnas de correo electrónico y edad
 - Método `ValidateRow` que establece el `ErrorText` de la fila y el error de la celda de fecha de cierre para los contactos `Closed`
 - Controlador `CellValidated` que borra los errores de celda obsoletos
 - Controlador `DataError` con `ThrowException = false`, un mensaje comprensible y la excepción registrada
 - Protección de `Save` que omite la fila nueva de marcador de posición y se niega a guardar mientras alguna fila tenga un error
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:

- La regla de correo electrónico de un compañero lee `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value` y «funciona en las pruebas». Describe la

edición exacta que se cuela y por qué.

- Tu regla de edad comprueba el rango de 0 a 120. ¿Qué debe hacer cuando el texto no es un número en absoluto, y por qué sigue haciendo falta un controlador `DataError` aunque la propia regla rechace «abc»?
- La regla `Closed` podría mostrarse como error de celda, como error de fila o como ambos. ¿Cuál has elegido y qué seguiría viendo un usuario que nunca pasa el ratón por encima de un icono?

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:

- ✓ La regla de correo electrónico de un compañero lee `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value` y «funciona en las pruebas». Describe la edición exacta que se cuela y por qué.
- ✓ Tu regla de edad comprueba el rango de 0 a 120. ¿Qué debe hacer cuando el texto no es un número en absoluto, y por qué sigue haciendo falta un controlador `DataError` aunque la propia regla rechace «abc»?
- ✓ La regla `Closed` podría mostrarse como error de celda, como error de fila o como ambos. ¿Cuál has elegido y qué seguiría viendo un usuario que nunca pasa el ratón por encima de un icono?

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 6 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 6 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Validation in Wisej.NET Course/Module 6`.



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.

- ✓ **Cambia una pieza sin mover las juntas.** *"Elige la parte de este proyecto ValidationClinic 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 ValidationClinic 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 ValidationClinic 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."*



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.