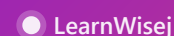


Ejercicio de programación con IA

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.



El mismo módulo, construido como de verdad vas a construir.

Todas las demás secciones de este módulo explican cómo funciona «Validación enlazada a datos y errores del modelo». Esta te pide que lo generes — dirigiendo a un asistente con la propia especificación del módulo y comprobando después que el resultado se ejecuta y hace lo que describe el enunciado del laboratorio. Es el flujo de trabajo que usarás en proyectos reales, practicado con algo cuya respuesta correcta ya conoces.

Qué estás pidiendo

Enlaza el formulario de alta de ValidationClinic a un ContactEditModel mediante un contactBindingSource y añade las anotaciones Required, StringLength, EmailAddress y Range a sus propiedades Name, Email y Age. En btnSave_Click llama primero a contactBindingSource.EndEdit(), ejecuta Validator.TryValidateObject con validateAllProperties establecido en true y asocia el nombre de miembro de cada ValidationResult a su control mediante un mapa de campos y el resumen de validación. Después implementa IDataErrorInfo en un segundo modelo, dirige el ErrorProvider al origen de enlace con BindToDataAndErrors y muestra cómo aparecen los mismos errores a través del enlace a datos sin ninguna llamada a SetError escrita a mano.

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

- ✓ ContactEditModel con DataAnnotations enlazado al formulario mediante un BindingSource
- ✓ Controlador de Save que llama a EndEdit antes de ejecutar Validator.TryValidateObject
- ✓ Mapa de campos que convierte los nombres de miembro de ValidationResult en errores de control y líneas del resumen
- ✓ Modelo IDataErrorInfo cuyos errores aparecen a través de un ErrorProvider enlazado a datos
- ✓ Llamada a BindToDataAndErrors usada cuando el origen de datos del proveedor cambia en tiempo de ejecución

Por qué tiene esta forma

Todo lo que ValidationClinic comprueba hasta ahora vive en el formulario de alta: un controlador Validating aquí, una regla del extensor Validation allá, una llamada a ContactValidator en btnSave_Click. Pero un contacto no solo llega a través de ese formulario. Puede editarse en la cuadrícula de contactos, importarse desde un archivo o crearse desde un servicio, y ninguno de esos caminos ejecuta los

controladores del formulario. Este módulo traslada las reglas básicas al propio `ContactEditModel`, para que los datos lleven su propia definición de «válido» y todas las pantallas puedan hacerles la misma pregunta. 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
<code>BindingSource</code>	Un componente que se sitúa entre los controles y tu objeto: su <code>DataSource</code> es el modelo, y cada control se enlaza a una propiedad a través de él en lugar de copiarse a mano.
<code>EndEdit()</code>	Lleva al modelo el valor que el usuario todavía está editando en el control, para que la validación lea lo que hay en pantalla y no el último valor confirmado.
<code>DataAnnotations</code>	Atributos como <code>[Required]</code> , <code>[StringLength]</code> , <code>[EmailAddress]</code> y <code>[Range]</code> colocados en las propiedades del modelo, cada uno con un <code>ErrorMessage</code> escrito para el usuario.
<code>Validator.TryValidateObject</code>	El método de .NET que ejecuta esos atributos sobre un objeto y rellena una lista de objetos <code>ValidationResult</code> ; no sabe nada de controles ni de Wisej.NET.
<code>MemberNames</code>	Los nombres de las propiedades a las que pertenece un <code>ValidationResult</code> , que son el único vínculo entre un mensaje de error y el control que debe mostrarlo.
<code>IDataErrorInfo</code>	Una interfaz que permite a un modelo responder «¿qué le pasa a esta propiedad?» mediante un índice, para que la capa de enlace pida los errores en lugar de que tu código los empuje.
ErrorProvider enlazado a datos	Un <code>ErrorProvider</code> cuyo <code>DataSource</code> es el mismo origen de enlace que el de los controles, de modo que muestra los errores del modelo junto a los controles enlazados sin ninguna llamada a <code>SetError</code> .

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 5. Puedes abrir tú mismo [module-05.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 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 5 – «Validación enlazada a datos y errores del modelo» – 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-05.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: Enlaza el formulario de alta de ValidationClinic a un ContactEditModel mediante un `contactBindingSource` y añade las anotaciones `Required`, `StringLength`, `EmailAddress` y `Range` a sus propiedades `Name`, `Email` y `Age`. En `btnSave_Click` llama primero a `contactBindingSource.EndEdit()`, ejecuta `Validator.TryValidateObject` con `validateAllProperties` establecido en `true` y asocia el nombre de miembro de cada `ValidationResult` a su control mediante un mapa de campos y el resumen de validación. Después implementa `IDataErrorInfo` en un segundo modelo, dirige el `ErrorProvider` al origen de enlace con `BindToDataAndErrors` y muestra cómo aparecen los mismos errores a través del enlace a datos sin ninguna llamada a `SetError` escrita a mano.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `ContactEditModel` con `DataAnnotations` enlazado al formulario mediante un `BindingSource`
 - Controlador de `Save` que llama a `EndEdit` antes de ejecutar `Validator.TryValidateObject`
 - Mapa de campos que convierte los nombres de miembro de `ValidationResult` en errores de control y líneas del resumen
 - Modelo `IDataErrorInfo` cuyos errores aparecen a través de un `ErrorProvider` enlazado a datos
 - Llamada a `BindToDataAndErrors` usada cuando el origen de datos del proveedor cambia en tiempo de ejecución
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 tester escribe un nuevo correo electrónico y pulsa Intro para guardar, y lo que se comprueba y almacena es el correo electrónico antiguo y válido. ¿Qué línea falta y por qué el formulario parece correcto en pantalla mientras el modelo está mal?
- Quitas `validateAllProperties: true` y todos los guardados siguen «funcionando». ¿Cuáles de tus cuatro anotaciones han dejado de ejecutarse y qué datos incorrectos pueden llegar ahora

al repositorio?

– El servicio de importación crea contactos sin abrir el formulario. ¿Qué reglas de este módulo siguen protegiendo ese camino y cuáles solo se ejecutan porque un usuario ha hecho clic en Save?

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 tester escribe un nuevo correo electrónico y pulsa Intro para guardar, y lo que se comprueba y almacena es el correo electrónico antiguo y válido. ¿Qué línea falta y por qué el formulario parece correcto en pantalla mientras el modelo está mal?
- ✓ Quitas `validateAllProperties: true` y todos los guardados siguen «funcionando». ¿Cuáles de tus cuatro anotaciones han dejado de ejecutarse y qué datos incorrectos pueden llegar ahora al repositorio?
- ✓ El servicio de importación crea contactos sin abrir el formulario. ¿Qué reglas de este módulo siguen protegiendo ese camino y cuáles solo se ejecutan porque un usuario ha hecho clic en Save?

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 5 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 5 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Validation in Wisej.NET Course/Module 5`.



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 ValidationClinic 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 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."*



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.