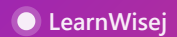


## Ejercicio de programación con IA

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 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

Completa el proyecto final de ValidationClinic. Escribe btnSave\_Click como un flujo fijo: restablece el resumen, llama a bindingSource.EndEdit(), detente si ValidateChildren() falla, detente si falla la comprobación de DataAnnotations, detente si ContactValidator devuelve errores de negocio, detente si la cuadrícula de contactos todavía tiene errores de fila y solo entonces guarda dentro de un try/catch que convierta un nombre duplicado en un error de campo y cualquier otra excepción en un mensaje comprensible. Crea SummaryErrorProvider implementando IErrorProvider, asígnalo a validation.ErrorProvider y encamina los errores a través de un método auxiliar SetError que escriba tanto en el ErrorProvider de iconos como en el proveedor de resumen. Prueba valores obligatorios, no válidos, válidos y límite, y escribe un breve README que explique el diseño de la validación.

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

- ✓ Flujo de Save que ejecuta EndEdit, ValidateChildren y las comprobaciones del modelo, de negocio y de la cuadrícula antes de guardar
- ✓ Clase SummaryErrorProvider que implementa SetError y GetError de IErrorProvider
- ✓ Método auxiliar SetError que escribe tanto en el ErrorProvider de iconos como en el proveedor de resumen
- ✓ try/catch alrededor del guardado que separa un error de nombre duplicado de los fallos del sistema
- ✓ README con el diseño de la validación y casos de prueba para valores obligatorios, no válidos, válidos y límite

## Por qué tiene esta forma

Un usuario que rellena el formulario de alta de ValidationClinic nunca pregunta qué API ha encontrado un error; solo se fija en si la pantalla le dice qué está mal y le deja terminar. Después de seis módulos, la aplicación tiene controladores de campo, reglas del extensor, un servicio validador, modelos con anotaciones y reglas de cuadrícula, pero cada elemento se añadió por separado y todavía nada garantiza que todos se ejecuten, en el orden correcto, cada vez que se hace clic en Save. Este módulo une las capas en un flujo de Save predecible y en un único resumen visible, y después te pide demostrar el diseño en un proyecto final.

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                                                                                                                                                                                                   |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flujo de Save                                | Una secuencia fija dentro de btnSave_Click que confirma las ediciones pendientes, ejecuta por turnos cada capa de validación y sale en la primera capa que falla, para que nada llegue al repositorio sin comprobar. |
| Salida anticipada                            | Cada etapa termina el controlador con return cuando encuentra errores, así que las etapas posteriores nunca se ejecutan sobre datos que una etapa anterior ya ha rechazado.                                          |
| IErrorProvider                               | La pequeña interfaz a la que el sistema de validación de Wisej.NET envía los mensajes: SetError(Control, string) para establecer o borrar un mensaje y GetError(Control) para volver a leerlo.                       |
| SummaryErrorProvider                         | Tu propio IErrorProvider, que guarda un diccionario de pares control-mensaje y reescribe a partir de él una etiqueta de resumen cada vez que se establece o se borra un error.                                       |
| Proveedores dobles                           | Un único método auxiliar SetError que escribe el mismo mensaje en el ErrorProvider integrado (el icono junto al campo) y en el proveedor de resumen (la lista de la parte superior).                                 |
| Error del usuario frente a fallo del sistema | Un error que el usuario puede corregir, como un nombre duplicado, se convierte en un mensaje de campo; una excepción inesperada se registra para los desarrolladores y se muestra como un mensaje genérico y sereno. |
| Prueba de límites                            | Un valor de prueba situado justo en el borde de una regla, como una edad de 0 o 120, que detecta los errores por uno que los valores válidos y no válidos corrientes no detectan.                                    |

## 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](https://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](https://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 7 – «UX de validación en producción, proveedores de errores personalizados y proyecto final» – 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-07.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: Completa el proyecto final de ValidationClinic. Escribe `btnSave_Click` como un flujo fijo: restablece el resumen, llama a `bindingSource.EndEdit()`, detente si `ValidateChildren()` falla, detente si falla la comprobación de `DataAnnotations`, detente si `ContactValidator` devuelve errores de negocio, detente si la cuadrícula de contactos todavía tiene errores de fila y solo entonces guarda dentro de un `try/catch` que convierta un nombre duplicado en un error de campo y cualquier otra excepción en un mensaje comprensible. Crea `SummaryErrorProvider` implementando `IErrorProvider`, asígnalo a `validation.ErrorProvider` y encamina los errores a través de un método auxiliar `SetError` que escriba tanto en el `ErrorProvider` de iconos como en el proveedor de resumen. Prueba valores obligatorios, no válidos, válidos y límite, y escribe un breve README que explique el diseño de la validación.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
  - Flujo de Save que ejecuta `EndEdit`, `ValidateChildren` y las comprobaciones del modelo, de negocio y de la cuadrícula antes de guardar
  - Clase `SummaryErrorProvider` que implementa `SetError` y `GetError` de `IErrorProvider`
  - Método auxiliar `SetError` que escribe tanto en el `ErrorProvider` de iconos como en el proveedor de resumen
  - `try/catch` alrededor del guardado que separa un error de nombre duplicado de los fallos del sistema
  - README con el diseño de la validación y casos de prueba para valores obligatorios, no válidos, válidos y límite
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 en el formulario y hace clic en Save inmediatamente; se guarda el correo electrónico antiguo. ¿Qué etapa del flujo falta o está

mal colocada, y por qué importa su posición?

- Tu `SummaryErrorProvider` muestra «Email is required.» dos veces después de que el usuario corrija el campo y lo vuelva a estropear. ¿Qué parte de `SetError` debería evitarlo y de qué depende?
- El repositorio lanza una excepción con un mensaje de la base de datos cuando se cae la red. Describe paso a paso qué ve el usuario y qué puede encontrar después el desarrollador, y explica por qué esto es distinto del caso del nombre duplicado.

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 en el formulario y hace clic en Save inmediatamente; se guarda el correo electrónico antiguo. ¿Qué etapa del flujo falta o está mal colocada, y por qué importa su posición?
- ✓ Tu `SummaryErrorProvider` muestra «Email is required.» dos veces después de que el usuario corrija el campo y lo vuelva a estropear. ¿Qué parte de `SetError` debería evitarlo y de qué depende?
- ✓ El repositorio lanza una excepción con un mensaje de la base de datos cuando se cae la red. Describe paso a paso qué ve el usuario y qué puede encontrar después el desarrollador, y explica por qué esto es distinto del caso del nombre duplicado.

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 `Validation in Wisej.NET 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.

- ✓ **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."*
- ✓ **Demuestra que de verdad funciona.** *"Recorre este proyecto entregable por entregable frente a la especificación del módulo en <https://learnwisej.com/assets/courses/validation-in-wisej-net/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."*



### 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.