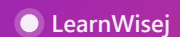


Ejercicio de programación con IA

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.



Escribir el código ya es la mitad fácil.

Conseguir que un asistente moderno genere código que funcione para «Validación personalizada: métodos, servicios y clases `ValidationRule`» solo requiere un buen prompt. Juzgarlo, corregirlo y ampliarlo requiere todo lo que ha cubierto este módulo. Ese es el reparto sobre el que se construye este ejercicio: tú aportas la especificación y la revisión; el asistente, el tecleo.

Qué estás pidiendo

Añade validación personalizada a `ValidationClinic` en tres pasos. Escribe `ValidateStartEndDates` como un método que devuelve una cadena vacía cuando todo es válido y un mensaje cuando la fecha de inicio es posterior a la fecha de fin, y establece ese mensaje tanto en `ctpStart` como en `ctpEnd`. Traslada las reglas del nombre, el correo electrónico y la fecha de nacimiento a un servicio `ContactValidator` que devuelve una lista de records `ValidationMessage`, y asocia cada nombre de campo a su control en `ApplyModelValidation`. Después crea `MinimumAgeValidationRule` derivando de `ValidationRule` con una propiedad `MinimumAge`, sobrescribe `OnValidating` para calcular la edad a partir del `DateTimePicker`, asóciala a `ctpBirthDate` mediante el extensor `Validation` y ejecuta el servicio en `btnSave_Click` antes de guardar nada.

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

- ✓ Método `ValidateStartEndDates` que establece el mismo error en los dos selectores de fecha y borra ambos cuando se corrige
- ✓ Servicio `ContactValidator` que devuelve records `ValidationMessage` para el nombre, el correo electrónico y la fecha de nacimiento
- ✓ Método `ApplyModelValidation` que asocia cada nombre de campo a su control y al resumen
- ✓ Clase `MinimumAgeValidationRule` que sobrescribe `OnValidating`, asociada a `ctpBirthDate`
- ✓ Controlador de `Save` que ejecuta las reglas de campo, después el servicio validador y solo entonces guarda

Por qué tiene esta forma

Después del Módulo 3, el formulario de alta de `ValidationClinic` sabe que un correo electrónico necesita una @ y que una edad debe ser un número entero. Pero sigue guardando sin problema un contacto cuya fecha de inicio es posterior a la fecha de fin, o una fecha de nacimiento que hace que la persona tenga

once años. Las reglas integradas comprueban la forma de un valor; las aplicaciones reales también necesitan reglas sobre el significado, y algunas de esas reglas implican a más de un control. Este módulo da a esas reglas un lugar adecuado, para que se escriban una sola vez, se muestren con claridad y nunca se omitan al guardar.

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
método de validación personalizado	Un método del formulario que responde a una única pregunta y devuelve una cadena vacía cuando los datos son válidos, o un mensaje claro cuando no lo son.
servicio de validación	Una clase como <code>ContactValidator</code> que comprueba un modelo sin tocar ningún control, de modo que las mismas reglas se ejecutan desde el formulario, una cuadrícula o una prueba unitaria.
record <code>ValidationMessage</code>	Un tipo de resultado mínimo que contiene un nombre de <code>Field</code> y un <code>Message</code> ; el servicio devuelve una lista de ellos en lugar de establecer los errores por sí mismo.
<code>ApplyModelValidation</code>	El método del formulario que borra los errores antiguos, asocia cada nombre de campo a su control con <code>errorProvider.SetError</code> , rellena el resumen e informa de si todo ha pasado.
<code>ValidationRule</code>	La clase base de Wisej.NET que hay detrás de cada regla del extensor <code>Validation</code> ; derivar de ella convierte tu propia comprobación en una regla que puedes asociar a cualquier control.
<code>OnValidating</code> / <code>OnValidated</code> / <code>OnControlCreated</code>	Las tres sobrescrituras de una clase de regla: comprobar el valor y devolver <code>true</code> o <code>false</code> , actuar después de que un valor haya pasado y preparar lo necesario una vez que el control existe en el navegador.
regla entre campos	Una regla cuya respuesta depende de dos o más controles, como que la fecha de inicio sea anterior a la de fin, así que su error corresponde a todos los controles implicados.

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 4. Puedes abrir tú mismo [module-04.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 4 – «Validación personalizada: métodos, servicios y clases ValidationRule» – 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-04.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: Añade validación personalizada a ValidationClinic en tres pasos. Escribe `ValidateStartEndDates` como un método que devuelve una cadena vacía cuando todo es válido y un mensaje cuando la fecha de inicio es posterior a la fecha de fin, y establece ese mensaje tanto en `ctpStart` como en `ctpEnd`. Traslada las reglas del nombre, el correo electrónico y la fecha de nacimiento a un servicio `ContactValidator` que devuelve una lista de records `ValidationMessage`, y asocia cada nombre de campo a su control en `ApplyModelValidation`. Después crea `MinimumAgeValidationRule` derivando de `ValidationRule` con una propiedad `MinimumAge`, sobrescribe `OnValidating` para calcular la edad a partir del `DateTimePicker`, asóciala a `ctpBirthDate` mediante el extensor `Validation` y ejecuta el servicio en `btnSave_Click` antes de guardar nada.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Método `ValidateStartEndDates` que establece el mismo error en los dos selectores de fecha y borra ambos cuando se corrige
 - Servicio `ContactValidator` que devuelve records `ValidationMessage` para el nombre, el correo electrónico y la fecha de nacimiento
 - Método `ApplyModelValidation` que asocia cada nombre de campo a su control y al resumen
 - Clase `MinimumAgeValidationRule` que sobrescribe `OnValidating`, asociada a `ctpBirthDate`
 - Controlador de `Save` que ejecuta las reglas de campo, después el servicio validador y solo entonces guarda
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 comprobación de la edad mínima podría estar en `ContactValidator` o en `MinimumAgeValidationRule`. ¿Qué percibe de forma distinta el usuario en cada caso y cuándo querías tener ambas?
- Tu controlador de `Save` llama a `ApplyModelValidation` después de establecer los errores de

fechas, y los iconos de las fechas nunca aparecen. Explica por qué y qué orden lo soluciona.
– ¿Cómo demostrarías que ContactValidator rechaza un correo electrónico sin @ sin iniciar la aplicación ni abrir un navegador?

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 comprobación de la edad mínima podría estar en ContactValidator o en MinimumAgeValidationRule. ¿Qué percibe de forma distinta el usuario en cada caso y cuándo querrías tener ambas?
- ✓ Tu controlador de Save llama a ApplyModelValidation después de establecer los errores de fechas, y los iconos de las fechas nunca aparecen. Explica por qué y qué orden lo soluciona.
- ✓ ¿Cómo demostrarías que ContactValidator rechaza un correo electrónico sin @ sin iniciar la aplicación ni abrir un navegador?

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 4 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 4 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Validation in Wisej.NET Course/Module 4`.



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.

- ✓ **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-04.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."*
- ✓ **Revisalo como un pull request.** *"Explícame 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."*



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.