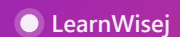


Ejercicio de programación con IA

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.



Instrúyelo, léelo y luego rómpelo.

Este ejercicio tiene tres partes, y solo la primera trata del prompt. Le das al asistente la especificación del módulo y recibes una solución ValidationClinic; lees lo que escribió y decides si de verdad hace lo que pedía el módulo; y después lo llevas más allá de lo que pedía el módulo. En la segunda y la tercera parte es donde se aprende.

Qué estás pidiendo

Sustituye los controladores de campo escritos a mano en ValidationClinic por el extensor Validation. Añade un componente Validation y un ErrorProvider al formulario de alta, establece ContainerControl y validation.ErrorProvider, y llama a SetValidationRules en un método ConfigureValidationRules: Required y después Email en txtEmail, Required y después Integer en txtAge, Required y después Telephone en txtPhone, Currency en txtCreditLimit y una regla Regex para el código de cliente. Coloca Required en primer lugar en todos los casos para que gane el mensaje más útil, engancha el evento Validating del extensor para actualizar el resumen y traslada los arrays de reglas a un pequeño método auxiliar reutilizable para que otro formulario pueda aplicar la misma configuración.

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

- ✓ Componente Validation conectado al formulario y a un ErrorProvider como destino de visualización
- ✓ Método ConfigureValidationRules que llama a SetValidationRules para el correo electrónico, la edad, el teléfono, el límite de crédito y el código de cliente
- ✓ Regla Required colocada en primer lugar en cada campo obligatorio, con un texto InvalidMessage claro
- ✓ Controlador Validating del extensor que mantiene el resumen de validación sincronizado con las reglas
- ✓ Método auxiliar de configuración de reglas reutilizable al que puede llamar un segundo formulario

Por qué tiene esta forma

Al final del Módulo 2 el formulario de alta de ValidationClinic funciona, pero fíjate en cuánto código ha hecho falta: un método auxiliar para el nombre, otro para el correo electrónico, otro para el teléfono, y cada uno repite las mismas comprobaciones de «¿está vacío?, ¿tiene la forma correcta?». Añade la edad, el límite de crédito y un código de cliente, y el formulario crece con otro puñado de métodos casi idénticos. Las comprobaciones no están mal; simplemente se vuelven a escribir en cada campo y en cada formulario. Este módulo sustituye esa repetición por el extensor Validation, con el que describes cómo es un campo válido y Wisej.NET ejecuta las comprobaciones por ti.

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
extensor Validation	Un componente no visual que añades a un formulario; se engancha a los eventos Validating y Validated de cada control que tiene reglas, así que ya no escribes un controlador por campo.
propiedad extendida	Una propiedad que un componente añade a otros controles; en cuanto el extensor está en el formulario, cada control compatible muestra una entrada ValidationRules en la ventana Propiedades.
SetValidationRules	El equivalente en código del diseñador: validation.SetValidationRules(txtEmail, new ValidationRule[] { ... }) asigna a un control su lista de reglas en tiempo de ejecución.
reglas integradas	Comprobaciones ya preparadas para entradas habituales: RequiredValidationRule, EmailValidationRule, IntegerValidationRule, DecimalValidationRule, CurrencyValidationRule, TelephoneValidationRule y RegexValidationRule.
orden de las reglas	Las reglas se ejecutan en el orden en que las enumeras; la primera que falla detiene las demás y cancela el evento Validating del control.
destino de visualización	Dónde aparece el InvalidMessage de la regla que falla: en el propio control, mediante un icono de ErrorProvider o como texto en una Label, todos accesibles a través de validation.ErrorProvider.
método auxiliar de configuración de reglas	Un pequeño método que construye los arrays de reglas de los campos habituales, para que un segundo formulario obtenga las mismas comprobaciones y la misma redacción con solo llamarlo.

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 3. Puedes abrir tú mismo [module-03.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 3 – «Reglas de validación integradas y el extensor Validation» – 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-03.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: Sustituye los controladores de campo escritos a mano en `ValidationClinic` por el extensor `Validation`. Añade un componente `Validation` y un `ErrorProvider` al formulario de alta, establece `ContainerControl` y `validation.ErrorProvider`, y llama a `SetValidationRules` en un método `ConfigureValidationRules`: `Required` y después `Email` en `txtEmail`, `Required` y después `Integer` en `txtAge`, `Required` y después `Telephone` en `txtPhone`, `Currency` en `txtCreditLimit` y una regla `Regex` para el código de cliente. Coloca `Required` en primer lugar en todos los casos para que gane el mensaje más útil, engancha el evento `Validating` del extensor para actualizar el resumen y traslada los arrays de reglas a un pequeño método auxiliar reutilizable para que otro formulario pueda aplicar la misma configuración.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Componente `Validation` conectado al formulario y a un `ErrorProvider` como destino de visualización
 - Método `ConfigureValidationRules` que llama a `SetValidationRules` para el correo electrónico, la edad, el teléfono, el límite de crédito y el código de cliente
 - Regla `Required` colocada en primer lugar en cada campo obligatorio, con un texto `InvalidMessage` claro
 - Controlador `Validating` del extensor que mantiene el resumen de validación sincronizado con las reglas
 - Método auxiliar de configuración de reglas reutilizable al que puede llamar un segundo formulario
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:

- Cambias el orden para que la regla `Email` se ejecute antes que `Required` en `txtEmail`. Describe exactamente qué ve un usuario después de dejar el campo vacío y por qué ese mensaje

es menos útil.

- Tanto los métodos auxiliares del Módulo 2 como el extensor terminan con un icono junto al campo. ¿Qué has dejado de escribir a mano y qué has tenido que seguir decidiendo tú?
- Un compañero quiere añadir «la fecha de fin debe ser posterior a la fecha de inicio» a ClinicRules. ¿Por qué el extensor no es el lugar adecuado para esa comprobación y cuándo dejaría de ejecutarse?

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:

- ✓ Cambias el orden para que la regla Email se ejecute antes que Required en txtEmail. Describe exactamente qué ve un usuario después de dejar el campo vacío y por qué ese mensaje es menos útil.
- ✓ Tanto los métodos auxiliares del Módulo 2 como el extensor terminan con un icono junto al campo. ¿Qué has dejado de escribir a mano y qué has tenido que seguir decidiendo tú?
- ✓ Un compañero quiere añadir «la fecha de fin debe ser posterior a la fecha de inicio» a ClinicRules. ¿Por qué el extensor no es el lugar adecuado para esa comprobación y cuándo dejaría de ejecutarse?

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 3 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 3 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Validation in Wisej.NET Course/Module 3`.



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 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-03.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 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."*



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.