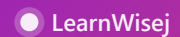


Esercizio di coding con l'IA

Costruisci il progetto ValidationClinic di questo modulo con ChatGPT o Claude partendo da un prompt già pronto che indirizza il modello alla specifica del modulo, poi rivedi, esegui ed estendi quello che ti restituisce — confrontandolo anche con la nostra build di riferimento su GitHub.



Istruiscilo, leggilo, poi mettilo alla prova.

Questo esercizio ha tre parti, e solo la prima riguarda il prompt. Istruisci un assistente con la specifica del modulo e ti fai restituire una soluzione ValidationClinic; leggi quello che ha scritto e decidi se fa davvero ciò che il modulo chiedeva; poi lo spingi più in là di quanto il modulo chiedesse. La seconda e la terza parte sono dove si impara.

Che cosa stai chiedendo

Sostituisci i gestori dei campi scritti a mano in ValidationClinic con l'extender Validation. Aggiungi un componente Validation e un ErrorProvider al form di accettazione, imposta ContainerControl e validation.ErrorProvider, poi chiama SetValidationRules in un metodo ConfigureValidationRules: Required e poi Email su txtEmail, Required e poi Integer su txtAge, Required e poi Telephone su txtPhone, Currency su txtCreditLimit e una regola Regex per il codice cliente. Metti Required per prima ovunque così vince il messaggio più utile, aggancia l'evento Validating dell'extender per aggiornare il riepilogo e sposta gli array di regole in un piccolo helper riutilizzabile così un altro form può applicare la stessa configurazione.

Il modulo chiede questi deliverable, e il prompt qui sotto li chiede all'assistente esattamente così:

- ✓ Componente Validation collegato al form e a un ErrorProvider come destinazione di visualizzazione
- ✓ Metodo ConfigureValidationRules che chiama SetValidationRules per email, età, telefono, fido e codice cliente
- ✓ Regola Required posta per prima su ogni campo obbligatorio, con testo InvalidMessage in linguaggio semplice
- ✓ Gestore Validating dell'extender che mantiene il riepilogo di validazione allineato alle regole
- ✓ Helper di configurazione delle regole riutilizzabile che un secondo form può chiamare

Perché è fatto così

Alla fine del Modulo 2 il form di accettazione di ValidationClinic funziona, ma guarda quanto codice ha richiesto: un helper per il nome, un helper per l'email, un helper per il telefono, e ognuno ripete le stesse verifiche "è vuoto, ha la forma giusta". Aggiungi età, fido e codice cliente e il form si arricchisce di un'altra manciata di metodi quasi identici. Le verifiche non sono sbagliate; vengono solo riscritte su ogni campo e

su ogni form. Questo modulo sostituisce quella ripetizione con l'extender Validation, in cui descrivi come deve essere un campo valido e Wisej.NET esegue le verifiche al posto tuo.

Questi sono i termini che l'assistente deve azzeccare. Se quello che ti restituisce li usa con leggerezza, non ha capito il modulo:

Termine	Che cosa significa qui
extender Validation	Un componente non visuale che aggiungi a un form; si aggancia agli eventi Validating e Validated di ogni controllo che ha delle regole, così non scrivi più un gestore per ogni campo.
proprietà estesa	Una proprietà che un componente aggiunge ad altri controlli; quando l'extender è sul form, ogni controllo compatibile mostra una voce ValidationRules nella finestra Properties.
SetValidationRules	L'equivalente in codice del designer: validation.SetValidationRules(txtEmail, new ValidationRule[] { ... }) assegna a un controllo il suo elenco di regole a runtime.
regole integrate	Verifiche già pronte per l'input più comune: RequiredValidationRule, EmailValidationRule, IntegerValidationRule, DecimalValidationRule, CurrencyValidationRule, TelephoneValidationRule e RegexValidationRule.
ordine delle regole	Le regole vengono eseguite nell'ordine in cui le elenchi; la prima che fallisce ferma le altre e annulla l'evento Validating del controllo.
destinazione di visualizzazione	Dove compare l' InvalidMessage della regola che fallisce: sul controllo stesso, tramite un'icona di ErrorProvider oppure come testo in una Label, tutte raggiunte tramite validation.ErrorProvider.
helper di configurazione delle regole	Un piccolo metodo che costruisce gli array di regole per i campi comuni, così un secondo form ottiene le stesse verifiche e la stessa formulazione semplicemente chiamandolo.

Costruiscilo — scegli il tuo assistente

Entrambi i pulsanti aprono una nuova scheda con lo **stesso prompt già pronto**. Il prompt indirizza il modello alla specifica di questo modulo — un unico file Markdown con la lezione, la traccia del lab e i deliverable — così lavora sul materiale vero invece di tirare a indovinare che cosa sia il Modulo 3. Puoi aprire tu stesso [module-03.md](#) per vedere esattamente che cosa leggerà.



Usa ChatGPT

Apri [chatgpt.com](#) con il prompt di questo modulo già caricato. Funziona con GPT-5 o



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece

Codex; qualsiasi piano che possa navigare il web recupererà la specifica.

incollarlo in un terminale dentro una cartella vuota.



Mandalo così com'è e ottieni un progetto, non un frammento

Il prompt si chiude chiedendo l'intera soluzione in un unico `.zip` con i passi per compilarla ed eseguirla. Scaricalo, apri la soluzione in Visual Studio, premi F5 e ValidationClinic si apre su `localhost`. Ti servono l'SDK .NET e il pacchetto NuGet `Wisej-4`; tutto il resto il prompt chiede all'assistente di scriverlo da sé.

Sto seguendo il Modulo 3 - "Regole di validazione integrate e l'extender Validation" - del corso LearnWisej "Validazione in Wisej.NET", e voglio costruire con te il progetto del modulo. Rispondimi in italiano.

Comincia leggendo la specifica del modulo. È un unico file Markdown (in inglese) che contiene l'intera lezione, la traccia del lab e i deliverable:

<https://learnwisej.com/assets/courses/validation-in-wisej-net/specs/module-03.md>

Poi costruisci un progetto Wisej.NET 4 completo che io possa aprire in Visual Studio:

1. Progetto "ValidationClinic", C#, net8.0 o successivo, pacchetto NuGet Wisej-4 4.1.x, host Kestrel che chiama app.UseWisej().
2. Costruisci quello che chiede il lab della specifica: Sostituisci i gestori dei campi scritti a mano in ValidationClinic con l'extender Validation. Aggiungi un componente Validation e un ErrorProvider al form di accettazione, imposta ContainerControl e validation.ErrorProvider, poi chiama SetValidationRules in un metodo ConfigureValidationRules: Required e poi Email su txtEmail, Required e poi Integer su txtAge, Required e poi Telephone su txtPhone, Currency su txtCreditLimit e una regola Regex per il codice cliente. Metti Required per prima ovunque così vince il messaggio più utile, aggancia l'evento Validating dell'extender per aggiornare il riepilogo e sposta gli array di regole in un piccolo helper riutilizzabile così un altro form può applicare la stessa configurazione.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Componente Validation collegato al form e a un ErrorProvider come destinazione di visualizzazione
 - Metodo ConfigureValidationRules che chiama SetValidationRules per email, età, telefono, fido e codice cliente
 - Regola Required posta per prima su ogni campo obbligatorio, con testo InvalidMessage in linguaggio semplice
 - Gestore Validating dell'extender che mantiene il riepilogo di validazione allineato alle regole
 - Helper di configurazione delle regole riutilizzabile che un secondo form può chiamare
4. Segui la specifica ovunque nomini una classe, una proprietà o un evento di Wisej.NET - usa esattamente quello. Dove lascia una scelta aperta, dimmi quale API hai scelto e perché.
5. Tieni l'autorità sul server: stato e validazione vivono in .NET, il browser disegna e riferisce. Non spostare mai una decisione in JavaScript solo perché lì è più comodo.
6. Rendi il risultato dimostrabile: devo poterlo eseguire e vedere il comportamento di cui parla questo modulo, compreso che cosa succede quando qualcosa va storto.

Prima di scrivere codice, riformula con parole tue che cosa costruirai e dove starà ogni pezzo, così posso correggere il progetto finché costa poco.

Quando il codice è pronto, impacchetta l'intera soluzione in un unico file .zip scaricabile e dammi i passi esatti per compilarla ed eseguirla.

Infine rispondi a queste domande della specifica, usando come prova la tua implementazione:

- Inverti l'ordine così la regola Email viene eseguita prima di Required su txtEmail. Descrivi esattamente che cosa vede un utente dopo aver lasciato il campo vuoto e perché quel messaggio è meno utile.
- Sia gli helper del Modulo 2 sia l'extender finiscono con un'icona accanto al campo. Che cosa hai smesso di scrivere a mano e che cosa hai dovuto comunque decidere tu?
- Un collega vuole aggiungere "la data di fine deve essere successiva alla data di inizio" a

ClinicRules. Perché l'extender è il posto sbagliato per quella verifica, e quando non verrebbe eseguita?

Chiedimi qualsiasi cosa non ti sia chiara prima di cominciare.

Poi rivedi quello che ti è tornato indietro

Codice che compila non è ancora codice giusto. Le domande interessanti sono domande di progetto: è questo il controllo Wisej.NET che il compito richiede davvero, segue il ciclo di vita che il framework ti offre, lo stato autorevole sta sul server o è stato parcheggiato nel browser perché era comodo? Leggi il risultato da revisore, non da cliente.

Questo modulo ti dà la checklist. Rispondi a ciascuna domanda guardando il codice che hai appena ricevuto:

- ✓ Inverti l'ordine così la regola Email viene eseguita prima di Required su txtEmail. Descrivi esattamente che cosa vede un utente dopo aver lasciato il campo vuoto e perché quel messaggio è meno utile.
- ✓ Sia gli helper del Modulo 2 sia l'extender finiscono con un'icona accanto al campo. Che cosa hai smesso di scrivere a mano e che cosa hai dovuto comunque decidere tu?
- ✓ Un collega vuole aggiungere "la data di fine deve essere successiva alla data di inizio" a ClinicRules. Perché l'extender è il posto sbagliato per quella verifica, e quando non verrebbe eseguita?

Tutto ciò che hai dovuto correggere vale la pena di scriverlo. Una breve nota con le convenzioni e le decisioni a cui sei arrivato, incollata nel primo messaggio della sessione successiva, vale più di qualsiasi prompt ingegnoso.



Questo l'abbiamo già costruito — confronta il tuo con il nostro

L'implementazione di riferimento del Modulo 3 si trova nel repository pubblico degli esempi. Costruisci prima il tuo, poi apri il nostro: la parte interessante sono le differenze.



[Apri l'esempio del Modulo 3 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Validation in Wisej.NET Course/Module 3`.



Il nostro esempio è un punto di partenza, non un prodotto finito

È stato costruito nello stesso modo in cui stai costruendo il tuo — con un prompt, poi corretto — e si ferma deliberatamente dove si ferma il modulo. Aspettati soluzioni provvisorie dove andrebbe codice di produzione, gestione degli errori essenziale e nessun test. Prendilo come un invito: usa quello che hai imparato in questo modulo, più l'assistente, per portarlo più avanti. Migliorare codice che non hai scritto tu è l'abilità che questo corso insegna davvero.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito — o sul nostro e prova questi. Ognuno è un esercizio di revisione ed estensione, non una riscrittura.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto ValidationClinic con un secondo caso che segue lo stesso schema del primo — un'altra entità, schermata o controllo della stessa forma, collegato allo stesso modo. Estrai in una classe base o in un servizio ciò che è davvero condiviso, ma solo dove se lo merita; due cose simili non chiedono sempre un'astrazione. Mostrami il diff e giustifica ogni cosa che hai estratto."*
- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/validation-in-wisej-net/specs/module-03.md> e mostrami la prova che ciascuno è davvero fatto — il percorso nel codice, su che cosa cliccare e che cosa dovrei vedere. Dove un deliverable è soddisfatto solo in parte, dillo chiaramente invece di arrampicarti sugli specchi. Poi aggiungi quello che manca."*
- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice ValidationClinic come se stessi rivedendo la pull request di un collega e non l'avessi scritto tu. Dove vive lo stato, che cosa succede dall'inizio alla fine su una singola azione dell'utente, che cosa rimanderesti indietro per modifiche, e quali parti non ti sentiresti tranquillo a mettere davanti a un cliente? Sii specifico e cita le righe di cui parli — non riassumermi il progetto."*



Portalo dentro il lab

Il lab pratico che chiude questo modulo chiede gli stessi deliverable che hai appena fatto produrre all'assistente. Usa quello che hai qui come forma della tua risposta e come prova a sostegno — non come la risposta in sé. Il lab viene valutato sul tuo ragionamento, e quella parte deve essere tua.