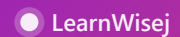


Esercizio di coding con l'IA

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.



Scrivere il codice è ormai la metà facile.

Ottenere codice funzionante per "Validazione personalizzata: metodi, servizi e classi ValidationRule" da un assistente moderno richiede un buon prompt. Giudicarlo, correggerlo ed estenderlo richiede tutto quello che questo modulo ha trattato. È su questa divisione che è costruito l'esercizio: tu metti la specifica e la revisione, l'assistente mette la digitazione.

Che cosa stai chiedendo

Aggiungi la validazione personalizzata a ValidationClinic in tre passi. Scrivi ValidateStartEndDates come metodo che restituisce una stringa vuota quando i dati sono validi e un messaggio quando la data di inizio è successiva alla data di fine, e imposta quel messaggio sia su dtpStart sia su dtpEnd. Sposta le regole su nome, email e data di nascita in un servizio ContactValidator che restituisce una lista di record ValidationMessage e riporta ogni nome di campo al suo controllo in ApplyModelValidation. Poi crea MinimumAgeValidationRule derivata da ValidationRule con una proprietà MinimumAge, fai l'override di OnValidating per calcolare l'età dal DateTimePicker, collegala a dtpBirthDate tramite l'extender Validation e esegui il servizio in btnSave_Click prima che venga salvato qualsiasi dato.

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

- ✓ Metodo ValidateStartEndDates che imposta lo stesso errore su entrambi i selettori di data e li cancella entrambi una volta corretti
- ✓ Servizio ContactValidator che restituisce record ValidationMessage per nome, email e data di nascita
- ✓ Metodo ApplyModelValidation che associa ogni nome di campo al suo controllo e al riepilogo
- ✓ Classe MinimumAgeValidationRule con override di OnValidating, collegata a dtpBirthDate
- ✓ Gestore di Save che esegue le regole dei campi, poi il servizio di validazione, e solo allora salva

Perché è fatto così

Dopo il Modulo 3 il form di accettazione di ValidationClinic sa che un'email ha bisogno di una @ e che l'età deve essere un numero intero. Però salva ancora senza problemi un contatto con la data di inizio successiva alla data di fine, o con una data di nascita che fa avere alla persona undici anni. Le regole integrate verificano la forma di un singolo valore; le applicazioni reali hanno bisogno anche di regole sul significato, e alcune di queste coinvolgono più di un controllo. Questo modulo dà a quelle regole una

collocazione adeguata, così vengono scritte una volta sola, mostrate con chiarezza e mai saltate al clic su Save.

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
metodo di validazione personalizzato	Un metodo del form che risponde a una sola domanda e restituisce una stringa vuota quando i dati sono validi, oppure un messaggio in linguaggio semplice quando non lo sono.
servizio di validazione	Una classe come <code>ContactValidator</code> che verifica un modello senza toccare alcun controllo, così le stesse regole vengono eseguite dal form, da una griglia o da uno unit test.
record <code>ValidationMessage</code>	Un piccolo tipo di risultato che contiene il nome di un campo (<code>Field</code>) e un <code>Message</code> ; il servizio ne restituisce una lista invece di impostare direttamente gli errori.
<code>ApplyModelValidation</code>	Il metodo del form che cancella gli errori precedenti, riporta ogni nome di campo al suo controllo con <code>errorProvider.SetError</code> , compila il riepilogo e indica se tutto è passato.
<code>ValidationRule</code>	La classe base di <code>Wisej.NET</code> dietro ogni regola dell'extender <code>Validation</code> ; derivando da essa trasformi la tua verifica in una regola che puoi collegare a qualsiasi controllo.
<code>OnValidating</code> / <code>OnValidated</code> / <code>OnControlCreated</code>	I tre override di una classe di regola: verificare il valore e restituire <code>true</code> o <code>false</code> , agire dopo che un valore è passato, e fare la preparazione una volta che il controllo esiste nel browser.
regola tra campi	Una regola la cui risposta dipende da due o più controlli, per esempio la data di inizio prima della data di fine, quindi il suo errore va su ogni controllo coinvolto.

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 4. Puoi aprire tu stesso [module-04.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 4 - "Validazione personalizzata: metodi, servizi e classi ValidationRule" - 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-04.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: Aggiungi la validazione personalizzata a ValidationClinic in tre passi. Scrivi ValidateStartEndDates come metodo che restituisce una stringa vuota quando i dati sono validi e un messaggio quando la data di inizio è successiva alla data di fine, e imposta quel messaggio sia su dtpStart sia su dtpEnd. Sposta le regole su nome, email e data di nascita in un servizio ContactValidator che restituisce una lista di record ValidationMessage e riporta ogni nome di campo al suo controllo in ApplyModelValidation. Poi crea MinimumAgeValidationRule derivata da ValidationRule con una proprietà MinimumAge, fai l'override di OnValidating per calcolare l'età dal DateTimePicker, collegala a dtpBirthDate tramite l'extender Validation ed esegui il servizio in btnSave_Click prima che venga salvato qualsiasi dato.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Metodo ValidateStartEndDates che imposta lo stesso errore su entrambi i selettori di data e li cancella entrambi una volta corretti
 - Servizio ContactValidator che restituisce record ValidationMessage per nome, email e data di nascita
 - Metodo ApplyModelValidation che associa ogni nome di campo al suo controllo e al riepilogo
 - Classe MinimumAgeValidationRule con override di OnValidating, collegata a dtpBirthDate
 - Gestore di Save che esegue le regole dei campi, poi il servizio di validazione, e solo allora salva
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:

- La verifica dell'età minima potrebbe stare in ContactValidator o in MinimumAgeValidationRule. Che cosa percepisce di diverso l'utente nei due casi, e quando vorresti averle entrambe?
- Il tuo gestore di Save chiama ApplyModelValidation dopo aver impostato gli errori sulle date, e le icone delle date non compaiono mai. Spiega perché, e quale ordine risolve il problema.

– Come dimostreresti che `ContactValidator` rifiuta un'email senza @ senza avviare l'app né aprire un browser?

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:

- ✓ La verifica dell'età minima potrebbe stare in `ContactValidator` o in `MinimumAgeValidationRule`. Che cosa percepisce di diverso l'utente nei due casi, e quando vorresti averle entrambe?
- ✓ Il tuo gestore di `Save` chiama `ApplyModelValidation` dopo aver impostato gli errori sulle date, e le icone delle date non compaiono mai. Spiega perché, e quale ordine risolve il problema.
- ✓ Come dimostreresti che `ContactValidator` rifiuta un'email senza @ senza avviare l'app né aprire un browser?

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 4 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 4 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Validation in Wisej.NET Course/Module 4`.



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.

- ✓ **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-04.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."*
- ✓ **Sostituisci un pezzo senza spostare le giunzioni.** *"Scegli la parte di questo progetto ValidationClinic che più evidentemente sta al posto di qualcosa di reale — una sorgente dati finta, un archivio in memoria, un servizio semplificato — e sostituiscila con un'implementazione di forma produttiva. Il contratto ai due lati non deve cambiare: stessi tipi, stesse chiamate, stesso comportamento in caso di errore. Dimmi che cosa l'implementazione reale ti ha costretto a gestire e che la versione provvisoria non gestiva."*



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.