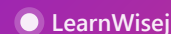


Esercizio di coding con l'IA

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.



Lo stesso modulo, costruito come costruirai davvero le cose.

Tutte le altre sezioni di questo modulo spiegano come funziona "Validazione con data binding ed errori del modello". Questa ti chiede di produrlo: dirigendo un assistente con la specifica del modulo stesso, poi verificando che il risultato giri e faccia quello che descrive la traccia del lab. È il flusso di lavoro che userai sul lavoro vero, provato su qualcosa di cui conosci già la risposta giusta.

Che cosa stai chiedendo

Collega il form di inserimento di ValidationClinic a un ContactEditModel tramite un contactBindingSource e aggiungi le annotazioni Required, StringLength, EmailAddress e Range alle sue proprietà Name, Email e Age. In btnSave_Click chiama per prima cosa contactBindingSource.EndEdit(), esegui Validator.TryValidateObject con validateAllProperties impostato a true e mappa il nome del membro di ogni ValidationResult sul suo controllo tramite una mappa dei campi e il riepilogo di validazione. Poi implementa IDataErrorInfo su un secondo modello, punta l'ErrorProvider alla binding source con BindToDataAndErrors e mostra gli stessi errori che compaiono tramite data binding senza una chiamata a SetError scritta a mano.

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

- ✓ ContactEditModel con DataAnnotations collegato al form tramite un BindingSource
- ✓ Gestore del salvataggio che chiama EndEdit prima di eseguire Validator.TryValidateObject
- ✓ Mappa dei campi che trasforma i nomi dei membri dei ValidationResult in errori dei controlli e righe del riepilogo
- ✓ Modello IDataErrorInfo i cui errori compaiono tramite un ErrorProvider con data binding
- ✓ Chiamata a BindToDataAndErrors usata quando la sorgente dati del provider cambia a runtime

Perché è fatto così

Tutto ciò che ValidationClinic verifica finora vive nel form di inserimento: qui un gestore Validating, là una regola sull'extender Validation, una chiamata a ContactValidator in btnSave_Click. Ma un contatto non arriva solo da quel form. Può essere modificato nella griglia dei contatti, importato da un file o creato da un servizio, e nessuno di questi percorsi esegue i gestori del form. Questo modulo sposta le regole di base sul ContactEditModel stesso, così che i dati portino con sé la propria definizione di "valido" e ogni schermata possa porre loro la stessa domanda.

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
BindingSource	Un componente che si colloca tra i controlli e il tuo oggetto: il suo DataSource è il modello, e ogni controllo si collega a una proprietà attraverso di esso invece di essere copiato a mano.
EndEdit()	Trasferisce nel modello il valore che l'utente sta ancora modificando nel controllo, così la validazione legge ciò che è a schermo e non l'ultimo valore confermato.
DataAnnotations	Attributi come [Required], [StringLength], [EmailAddress] e [Range] applicati alle proprietà del modello, ciascuno con un ErrorMessage scritto per l'utente.
Validator.TryValidateObject	Il metodo.NET che esegue quegli attributi su un oggetto e riempie una lista di oggetti ValidationResult; non sa nulla di controlli né di Wisej.NET.
MemberNames	I nomi delle proprietà a cui appartiene un ValidationResult, che sono l'unico collegamento tra un messaggio di errore e il controllo che dovrebbe mostrarlo.
IDataErrorInfo	Un'interfaccia che permette a un modello di rispondere a "che cosa non va in questa proprietà?" tramite un indicizzatore, così il livello di binding può chiedere gli errori invece di essere il tuo codice a spingerli.
ErrorProvider con data binding	Un ErrorProvider il cui DataSource è la stessa binding source dei controlli, così mostra gli errori del modello accanto ai controlli collegati senza alcuna chiamata a SetLastError.

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 5. Puoi aprire tu stesso [module-05.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 Codex; qualsiasi piano che possa navigare il web recupererà la specifica.



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece 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 5 - "Validazione con data binding ed errori del modello" - 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-05.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: Collega il form di inserimento di ValidationClinic a un ContactEditModel tramite un contactBindingSource e aggiungi le annotazioni Required, StringLength, EmailAddress e Range alle sue proprietà Name, Email e Age. In btnSave_Click chiama per prima cosa contactBindingSource.EndEdit(), esegui Validator.TryValidateObject con validateAllProperties impostato a true e mappa il nome del membro di ogni ValidationResult sul suo controllo tramite una mappa dei campi e il riepilogo di validazione. Poi implementa IDataErrorInfo su un secondo modello, punta l'ErrorProvider alla binding source con BindToDataAndErrors e mostra gli stessi errori che compaiono tramite data binding senza una chiamata a SetError scritta a mano.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - ContactEditModel con DataAnnotations collegato al form tramite un BindingSource
 - Gestore del salvataggio che chiama EndEdit prima di eseguire Validator.TryValidateObject
 - Mappa dei campi che trasforma i nomi dei membri dei ValidationResult in errori dei controlli e righe del riepilogo
 - Modello IDataErrorInfo i cui errori compaiono tramite un ErrorProvider con data binding
 - Chiamata a BindToDataAndErrors usata quando la sorgente dati del provider cambia a runtime
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:

- Un tester digita una nuova email e preme Invio per salvare, e a essere verificata e memorizzata è la vecchia email valida. Quale riga manca, e perché il form a schermo sembra corretto mentre il modello è sbagliato?
- Rimuovi validateAllProperties: true e ogni salvataggio "funziona" ancora. Quali delle tue quattro annotazioni hanno smesso di essere eseguite, e quali dati errati possono ora raggiungere il repository?
- Il servizio di importazione crea contatti senza aprire il form. Quali regole di questo modulo proteggono ancora quel percorso, e quali vengono eseguite solo perché un utente ha

fatto clic su Save?

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:

- ✓ Un tester digita una nuova email e preme Invio per salvare, e a essere verificata e memorizzata è la vecchia email valida. Quale riga manca, e perché il form a schermo sembra corretto mentre il modello è sbagliato?
- ✓ Rimuovi `validateAllProperties: true` e ogni salvataggio "funziona" ancora. Quali delle tue quattro annotazioni hanno smesso di essere eseguite, e quali dati errati possono ora raggiungere il repository?
- ✓ Il servizio di importazione crea contatti senza aprire il form. Quali regole di questo modulo proteggono ancora quel percorso, e quali vengono eseguite solo perché un utente ha fatto clic su Save?

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 5 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 5 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Validation in Wisej.NET Course/Module 5`.



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.

- ✓ **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."*
- ✓ **Fallo sopravvivere a una brutta giornata.** *"Esamina questo progetto ValidationClinic cercando le modalità di guasto che non gestisce ancora — input sbagliato o mancante, una dipendenza lenta o non disponibile, lavoro che arriva dopo che l'utente ha cambiato pagina, e due azioni che si accavallano. Per ciascuna dimmi prima se è un rischio reale in Wisej.NET 4 e perché, e solo dopo scrivi la correzione. Dove decidi che non serve gestire nulla, dillo e spiega il ragionamento invece di aggiungere codice difensivo."*



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.