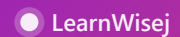


Esercizio di coding con l'IA

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.



L'assistente è la coppia di mani veloci. L'architetto resti tu.

Un assistente può produrre il progetto di questo modulo in pochi minuti. Quello che non può fare è decidere se il risultato è il progetto giusto per la tua applicazione: quella scelta resta tua. Il giudizio lo metti tu: istruiscilo per bene, poi verifica quello che ti torna indietro rispetto a ciò che il Modulo 7 ti ha appena insegnato.

Che cosa stai chiedendo

Completa il capstone ValidationClinic. Scrivi btnSave_Click come una pipeline fissa: azzerà il riepilogo, chiama bindingSource.EndEdit(), fermati se ValidateChildren() fallisce, fermati se la verifica DataAnnotations fallisce, fermati se ContactValidator restituisce errori di business, fermati se la griglia dei contatti ha ancora errori di riga, e solo allora salva dentro un try/catch che trasforma un nome duplicato in un errore di campo e qualsiasi altra eccezione in un messaggio comprensibile. Crea SummaryErrorProvider che implementa IErrorProvider, assegnalo a validation.ErrorProvider e instrada gli errori attraverso un metodo helper SetError che scrive sia nell'ErrorProvider a icona sia nel provider di riepilogo. Testa i casi obbligatorio, non valido, valido e valore limite e scrivi un breve README che spieghi il design della validazione.

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

- ✓ Pipeline di salvataggio che esegue EndEdit, ValidateChildren e le verifiche di modello, di business e della griglia prima di salvare
- ✓ Classe SummaryErrorProvider che implementa SetError e GetError di IErrorProvider
- ✓ Metodo helper SetError che scrive sia nell'ErrorProvider a icona sia nel provider di riepilogo
- ✓ try/catch attorno al salvataggio che separa l'errore di nome duplicato dagli errori di sistema
- ✓ README con il design della validazione e i casi di test obbligatorio, non valido, valido e valore limite

Perché è fatto così

Un utente che compila il form di accettazione di ValidationClinic non si chiede mai quale API abbia trovato un errore; nota solo se la schermata gli dice che cosa non va e gli permette di finire. Dopo sei moduli l'app ha gestori dei campi, regole dell'extender, un servizio di validazione, modelli annotati e regole della griglia, ma ognuno è stato aggiunto per conto suo, e niente garantisce ancora che vengano eseguiti tutti,

nell'ordine giusto, ogni volta che si fa clic su Save. Questo modulo unisce i livelli in un'unica pipeline di salvataggio prevedibile e in un unico riepilogo visibile, poi ti chiede di dimostrare il design in un capstone. 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
Pipeline di salvataggio	Una sequenza fissa dentro btnSave_Click che conferma le modifiche in sospeso, esegue a turno ogni livello di validazione e termina al primo livello che fallisce, così nulla arriva al repository senza essere verificato.
Uscita anticipata	Ogni fase chiude il gestore con return quando trova errori, così le fasi successive non vengono mai eseguite su dati che una fase precedente ha già rifiutato.
IErrorProvider	La piccola interfaccia a cui il sistema di validazione di Wisej.NET invia i messaggi: SetError(Control, string) per impostare o cancellare un messaggio e GetError(Control) per rileggerlo.
SummaryErrorProvider	Il tuo IErrorProvider che mantiene un dizionario di coppie controllo-messaggio e riscrive da esso una label di riepilogo ogni volta che un errore viene impostato o cancellato.
Doppio provider	Un unico metodo helper SetError che scrive lo stesso messaggio nell' ErrorProvider integrato (l'icona accanto al campo) e nel provider di riepilogo (l'elenco in alto).
Errore dell'utente o errore di sistema	Uno sbaglio che l'utente può correggere, come un nome duplicato, diventa un messaggio sul campo; un'eccezione imprevista viene registrata nel log per gli sviluppatori e mostrata come un messaggio generico e tranquillo.
Test del valore limite	Un valore di test che sta esattamente sul bordo di una regola, come un'età di 0 o 120, e che scopre gli errori off-by-one che i normali valori validi e non validi non colgono.

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 7. Puoi aprire tu stesso [module-07.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 7 - "UX di validazione in produzione, error provider personalizzati e capstone" - 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-07.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: Completa il capstone ValidationClinic. Scrivi btnSave_Click come una pipeline fissa: azzera il riepilogo, chiama bindingSource.EndEdit(), fermati se ValidateChildren() fallisce, fermati se la verifica DataAnnotations fallisce, fermati se ContactValidator restituisce errori di business, fermati se la griglia dei contatti ha ancora errori di riga, e solo allora salva dentro un try/catch che trasforma un nome duplicato in un errore di campo e qualsiasi altra eccezione in un messaggio comprensibile. Crea SummaryErrorProvider che implementa IErrorProvider, assegnalo a validation.ErrorProvider e instrada gli errori attraverso un metodo helper SetError che scrive sia nell'ErrorProvider a icona sia nel provider di riepilogo. Testa i casi obbligatorio, non valido, valido e valore limite e scrivi un breve README che spieghi il design della validazione.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Pipeline di salvataggio che esegue EndEdit, ValidateChildren e le verifiche di modello, di business e della griglia prima di salvare
 - Classe SummaryErrorProvider che implementa SetError e GetError di IErrorProvider
 - Metodo helper SetError che scrive sia nell'ErrorProvider a icona sia nel provider di riepilogo
 - try/catch attorno al salvataggio che separa l'errore di nome duplicato dagli errori di sistema
 - README con il design della validazione e i casi di test obbligatorio, non valido, valido e valore limite
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 nel form e fa subito clic su Save; viene salvata la vecchia email. Quale fase della pipeline manca o è nel posto sbagliato, e perché la sua posizione è importante?
- Il tuo SummaryErrorProvider mostra due volte "L'email è obbligatoria." dopo che l'utente corregge il campo e poi lo rende di nuovo non valido. Quale parte di SetError dovrebbe

impedirlo, e da che cosa dipende?

– Quando la rete cade, il repository genera un'eccezione con un messaggio del database. Descrivi passo per passo che cosa vede l'utente e che cosa può trovare lo sviluppatore in seguito, e spiega perché questo caso è diverso da quello del nome duplicato.

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 nel form e fa subito clic su Save; viene salvata la vecchia email. Quale fase della pipeline manca o è nel posto sbagliato, e perché la sua posizione è importante?
- ✓ Il tuo SummaryErrorProvider mostra due volte "L'email è obbligatoria." dopo che l'utente corregge il campo e poi lo rende di nuovo non valido. Quale parte di SetLastError dovrebbe impedirlo, e da che cosa dipende?
- ✓ Quando la rete cade, il repository genera un'eccezione con un messaggio del database. Descrivi passo per passo che cosa vede l'utente e che cosa può trovare lo sviluppatore in seguito, e spiega perché questo caso è diverso da quello del nome duplicato.

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 7 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 7 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Validation in Wisej.NET Course/Module 7`.



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.

- ✓ **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."*
- ✓ **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-07.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."*



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.