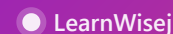


Esercizio di coding con l'IA

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.



Questo progetto non lo scriverai a mano. Lo dirigerai.

Quasi nessuno scrive più codice Wisej.NET da un file vuoto: un assistente scrive la prima bozza e lo sviluppatore decide se è giusta. Funziona però solo se conosci la materia, ed è quello che ti ha dato il resto di questo modulo. Qui affidi quelle conoscenze a ChatGPT o a Claude, ti fai restituire il progetto del modulo come soluzione Visual Studio funzionante, e lo rivedi come rivedresti la pull request di un collega.

Che cosa stai chiedendo

Fai in modo che la griglia dei contatti di ValidationClinic rifiuti i dati errati. Gestisci gridContacts.CellValidating: leggi e.FormattedValue invece del valore della cella, cancella l'ErrorText della cella, poi rifiuta un valore di EmailColumn vuoto o senza @ e un valore di AgeColumn fuori dall'intervallo da 0 a 120 con e.Cancel = true e un ErrorText di cella. Aggiungi un metodo ValidateRow che imposta l'ErrorText di riga e l'errore sulla cella ClosedDateColumn quando lo stato è Closed senza data di chiusura, cancella gli errori superati in CellValidated e gestisci DataError con ThrowException = false, un messaggio comprensibile sulla cella e un log di debug dell'eccezione. Salta la riga nuova segnaposto finché non contiene dati dell'utente, e blocca Save finché una qualsiasi riga ha ancora un errore.

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

- ✓ Gestore CellValidating che legge e.FormattedValue e valida le colonne dell'email e dell'età
- ✓ Metodo ValidateRow che imposta l'ErrorText di riga e l'errore sulla cella della data di chiusura per i contatti Closed
- ✓ Gestore CellValidated che cancella gli errori di cella superati
- ✓ Gestore DataError con ThrowException = false, un messaggio comprensibile e un'eccezione registrata nel log
- ✓ Controllo su Save che salta la riga nuova segnaposto e rifiuta di salvare finché una qualsiasi riga ha un errore

Perché è fatto così

I moduli dall'1 al 5 hanno protetto il form di accettazione di ValidationClinic, un contatto alla volta. La griglia dei contatti è diversa: un utente può spostarsi con Tab tra venti righe, scrivere un'email senza @, inserire "abc" come età e segnare un contatto come Closed senza data di chiusura, tutto senza aprire un form. Una cella della griglia non è un TextBox, quindi i gestori Validating e l' ErrorProvider che hai già

scritto non la raggiungono. Questo modulo dà alla griglia le sue regole di campo, le sue regole di riga e le sue regole sull'origine dati, e fa in modo che nessuna di esse trasformi un normale errore di digitazione in un crash o in una schermata bloccata.

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
CellValidating	L'evento della griglia che scatta quando la cella corrente sta per perdere il focus; è la versione per la griglia dell'evento Validating di un controllo e il punto in cui rifiutare un singolo valore.
e.FormattedValue	Il valore esattamente come l'utente l'ha digitato nell'editor, prima che venga confermato nella cella o nell'origine dati; leggere invece il Value della cella ti restituisce il valore vecchio.
e.Cancel	Impostarlo a true rifiuta di concludere la modifica, così il cursore resta nella cella errata finché l'utente non la corregge o annulla la modifica.
CellValidated	Scatta dopo che una cella ha superato la validazione; è un punto sicuro per cancellare un errore che non vale più o aggiornare un riepilogo.
ErrorText di cella	Un messaggio su una sola cella, mostrato con un'icona di errore dentro quella cella; usalo per un problema su un singolo valore.
ErrorText di riga	Un messaggio sull'intera riga, mostrato nell'intestazione di riga; usalo per una regola che dipende da più celle della stessa riga.
DataError	Viene generato quando la griglia non riesce a convertire, formattare o confermare un valore; con <code>e.ThrowException = false</code> sostituisci l'eccezione con un messaggio comprensibile.

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 6. Puoi aprire tu stesso [module-06.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 6 - "Validazione delle celle del DataGridView ed errori di riga" - 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-06.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: Fai in modo che la griglia dei contatti di ValidationClinic rifiuti i dati errati. Gestisci gridContacts.CellValidating: leggi e.FormattedValue invece del valore della cella, cancella l'ErrorText della cella, poi rifiuta un valore di EmailColumn vuoto o senza @ e un valore di AgeColumn fuori dall'intervallo da 0 a 120 con e.Cancel = true e un ErrorText di cella. Aggiungi un metodo ValidateRow che imposta l'ErrorText di riga e l'errore sulla cella ClosedDateColumn quando lo stato è Closed senza data di chiusura, cancella gli errori superati in CellValidated e gestisci DataError con ThrowException = false, un messaggio comprensibile sulla cella e un log di debug dell'eccezione. Salta la riga nuova segnaposto finché non contiene dati dell'utente, e blocca Save finché una qualsiasi riga ha ancora un errore.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Gestore CellValidating che legge e.FormattedValue e valida le colonne dell'email e dell'età
 - Metodo ValidateRow che imposta l'ErrorText di riga e l'errore sulla cella della data di chiusura per i contatti Closed
 - Gestore CellValidated che cancella gli errori di cella superati
 - Gestore DataError con ThrowException = false, un messaggio comprensibile e un'eccezione registrata nel log
 - Controllo su Save che salta la riga nuova segnaposto e rifiuta di salvare finché una qualsiasi riga ha un errore
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 regola sull'email di un compagno di corso legge grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value e "nei test funziona". Descrivi la modifica esatta che passa inosservata, e perché.
- La tua regola sull'età verifica l'intervallo da 0 a 120. Che cosa dovrebbe fare quando il testo non è affatto un numero, e perché serve comunque un gestore DataError anche se la regola stessa rifiuta "abc"?

- La regola Closed potrebbe essere mostrata come errore di cella, come errore di riga o come entrambi. Che cosa hai scelto, e che cosa vedrebbe comunque un utente che non passa mai il mouse sopra un'icona?

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 regola sull'email di un compagno di corso legge `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value` e "nei test funziona". Descrivi la modifica esatta che passa inosservata, e perché.
- ✓ La tua regola sull'età verifica l'intervallo da 0 a 120. Che cosa dovrebbe fare quando il testo non è affatto un numero, e perché serve comunque un gestore `DataError` anche se la regola stessa rifiuta "abc"?
- ✓ La regola Closed potrebbe essere mostrata come errore di cella, come errore di riga o come entrambi. Che cosa hai scelto, e che cosa vedrebbe comunque un utente che non passa mai il mouse sopra un'icona?

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 6 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 6 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Validation in Wisej.NET Course/Module 6`.



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.

- ✓ **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."*
- ✓ **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."*



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.