

Esercizio di coding con l'IA

Costruisci il progetto OperationsConsole 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 "Padroneggiare la DataGridView". 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

Costruisci la griglia Orders di Operations Console: una DataGridView collegata a un BindingSource con colonne esplicite e AutoGenerateColumns sotto controllo, una cella di stato resa con AllowHtml e CellFormatting, una colonna con editor personalizzato o comando, una versione in modalità virtuale o con cache per il percorso a grandi volumi, e una striscia di filtri e di stato composta dentro o intorno alla griglia.

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

- ✓ DataGridView Orders con colonne esplicite collegate a un BindingSource
- ✓ Cella di stato resa con AllowHtml e CellFormatting
- ✓ Una colonna con editor personalizzato o comando
- ✓ Versione della griglia in modalità virtuale o con cache per grandi volumi
- ✓ Striscia di filtri e di stato composta dentro o intorno alla griglia

Perché è fatto così

La DataGridView è il controllo che più di ogni altro decide se un'applicazione gestionale sembra matura. Ordini, fatture, scadenze, log, utenti e documenti finiscono tutti in una griglia, e gli utenti giudicano l'applicazione da come quella griglia carica, scorre, si modifica e riporta lo stato. Wisej.NET offre tutto, dal popolamento manuale alla modalità virtuale al disegno delle celle; la padronanza sta nello scegliere la tecnica giusta per la quantità di dati e per il compito dell'utente.

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 intermediario lato server tra le righe del view model e la griglia, che le dà una sorgente dati stabile, la nozione di riga corrente e l'ordinamento senza accoppiare la schermata al database.
AutoGenerateColumns	Crea una colonna per ogni proprietà pubblica del tipo in binding; comodo per un prototipo, pericoloso in produzione perché una modifica al modello riorganizza la schermata senza dire niente.
colonna esplicita	Una colonna che definisci tu con binding al campo, testo dell'intestazione, larghezza, allineamento, formato, sola lettura, modalità di ordinamento e tipo di editor, così l'interfaccia resta stabile quando il modello cambia.
modalità virtuale	Con VirtualMode attiva, la griglia conosce solo il numero di righe e chiede ogni valore visibile tramite CellValueNeeded: non viene creato nulla per le righe che nessuno vede.
cache lato server	Un buffer grande quanto una pagina che alimenta CellValueNeeded, così lo scorrimento non colpisce il database per ogni cella; l'evento DataRead è il punto documentato per costruirlo e mantenerlo.
editor di cella personalizzato	Un controllo assegnato alla proprietà Editor di una colonna; deve riportare il proprio valore nella cella, all'occorrenza tramite CellBeginEdit e CellEndEdit se non ha un evento di cambiamento adatto.
HTML nella cella o controllo nella cella	L'HTML prodotto in CellFormatting con AllowHtml è markup di sola visualizzazione e costa quasi nulla; un controllo incorporato è un vero controllo server più un widget nel browser, con un costo per ogni cella.

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 OperationsConsole 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 - "Padroneggiare la DataGridView" - del corso LearnWisej "Padroneggiare la libreria di controlli", 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/mastering-the-control-library/specs/module-05.md>

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

1. Progetto "OperationsConsole", 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: Costruisci la griglia Orders di Operations Console: una DataGridView collegata a un BindingSource con colonne esplicite e AutoGenerateColumns sotto controllo, una cella di stato resa con AllowHtml e CellFormatting, una colonna con editor personalizzato o comando, una versione in modalità virtuale o con cache per il percorso a grandi volumi, e una striscia di filtri e di stato composta dentro o intorno alla griglia.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - DataGridView Orders con colonne esplicite collegate a un BindingSource
 - Cella di stato resa con AllowHtml e CellFormatting
 - Una colonna con editor personalizzato o comando
 - Versione della griglia in modalità virtuale o con cache per grandi volumi
 - Striscia di filtri e di stato composta dentro o intorno alla griglia
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:

- Se il prossimo sprint OrderRow guadagna due nuove proprietà, che cosa cambia nella schermata con le colonne generate, e che cosa cambia con le tue colonne esplicite?
- Il tuo gestore di CellValueNeeded funziona con 200 ordini. Che cosa succede al database quando un utente trascina la barra di scorrimento attraverso 200.000 righe, e quale parte del tuo progetto assorbe il colpo?
- Perché il badge dello stato va in CellFormatting e non nel modello OrderRow, e che cosa andrebbe storto se il nome di un cliente finisse in quell'HTML senza essere codificato?

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:

- ✓ Se il prossimo sprint OrderRow guadagna due nuove proprietà, che cosa cambia nella schermata con le colonne generate, e che cosa cambia con le tue colonne esplicite?
- ✓ Il tuo gestore di CellValueNeeded funziona con 200 ordini. Che cosa succede al database quando un utente trascina la barra di scorrimento attraverso 200.000 righe, e quale parte del tuo progetto assorbe il colpo?
- ✓ Perché il badge dello stato va in CellFormatting e non nel modello OrderRow, e che cosa andrebbe storto se il nome di un cliente finisse in quell'HTML senza essere codificato?

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 `Mastering the Control Library 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.

- ✓ **Sostituisci un pezzo senza spostare le giunzioni.** *"Scegli la parte di questo progetto OperationsConsole 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 OperationsConsole 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 OperationsConsole 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.