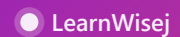


Esercizio di coding con l'IA

Costruisci il progetto VisualOperationsStudio 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.



Istruiscilo, leggilo, poi mettilo alla prova.

Questo esercizio ha tre parti, e solo la prima riguarda il prompt. Istruisci un assistente con la specifica del modulo e ti fai restituire una soluzione VisualOperationsStudio; leggi quello che ha scritto e decidi se fa davvero ciò che il modulo chiedeva; poi lo spingi più in là di quanto il modulo chiedesse. La seconda e la terza parte sono dove si impara.

Che cosa stai chiedendo

Dai a VisualOperationsStudio una griglia di operations con due colonne disegnate a mano. Aggiungi colonne esplicite a gridOperations, imposta UserPaint = true solo sulle colonne Health e Trend e gestisci CellPaint una volta sola: esci immediatamente quando e.RowIndex è negativo o e.ColumnIndex non è una delle due colonne bersaglio, poi disegna una barra di avanzamento colorata per soglia per Health e una sparkline a polilinea per Trend dentro e.ClipRectangle. Leggi i valori di ogni riga dall'oggetto MachineStatus associato invece di cercare dentro una lista nell'handler, rispetta i colori della riga selezionata e tieni l'handler privo di effetti collaterali. Associa 1.000 righe generate, scorri la griglia e poi costruisci lo stesso indicatore di stato una seconda volta come cella AllowHtml, così da poter dichiarare per iscritto quale dei due manderesti in produzione.

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

- ✓ gridOperations con colonne esplicite e UserPaint = true solo sulle colonne Health e Trend
- ✓ Un unico handler di CellPaint che verifica e.RowIndex e e.ColumnIndex prima di disegnare qualsiasi cosa
- ✓ Cella di avanzamento colorata per soglia e cella sparkline a polilinea disegnate dentro e.ClipRectangle
- ✓ Valori di riga letti dall'oggetto MachineStatus associato, senza ricerche, allocazioni o creazione di controlli per cella
- ✓ Una colonna di stato AllowHtml costruita in parallelo, più una nota che confronta le due soluzioni su uno scorrimento di 1.000 righe

Perché è fatto così

Il Modulo 2 ha dato a VisualOperationsStudio un TelemetryGauge: un controllo, una macchina, qualche centinaio di ridisegni all'ora. La griglia di operations è un animale diverso. Un capoturno apre una lista di

mille macchine e vuole vedere a colpo d'occhio quali sono vicine al limite e quali sono andate alla deriva per tutta la mattina. Dei numeri in colonna non bastano, e un gauge per riga sarebbe mille controlli lato server in una sola sessione. È la griglia stessa che deve disegnare, una volta per ogni cella visibile, e smettere nel momento in cui la cella esce dallo scorrimento.

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
UserPaint	L'interruttore esplicito: <code>DataGridViewCell.UserPaint = true</code> per una singola cella, <code>DataGridViewColumn.UserPaint = true</code> per un'intera colonna. Senza di esso la griglia non ti chiederà mai di disegnare.
CellPaint	L'evento a livello di griglia sollevato per le celle disegnate a mano quando devono essere rese; ti iscrivi una volta sola sulla griglia, non una per riga, e smisti dentro l'handler.
<code>DataGridViewCellPaintEventArgs</code>	Deriva da <code>PaintEventArgs</code> , quindi porta <code>Graphics</code> e <code>ClipRectangle</code> , e aggiunge <code>RowIndex</code> e <code>ColumnIndex</code> così che un solo handler possa servire più colonne.
niente helper WinForms	Gli argomenti dell'evento documentati da Wisej.NET non espongono i metodi di comodo di WinForms <code>PaintBackground()</code> e <code>PaintContent()</code> . È il tuo handler a possedere l'intero risultato visivo di quella cella.
<code>e.ClipRectangle</code> come superficie	Il rettangolo di destinazione locale della cella. Le colonne possono essere ridimensionate e la griglia disposta in modo responsive, quindi ogni coordinata che disegni va ricavata da lì e non da una costante.
helper di rendering puri	Metodi con la forma <code>DrawProgressCell(Graphics g, Rectangle rect, double value)</code> , così l'handler si riduce a codice di instradamento e lo stesso helper potrà poi rendere dentro un bitmap off-screen.
contenuto di cella <code>AllowHtml</code>	L'alternativa nativa del browser: icone, span, label e CSS resi dal client, ed è la ragione per cui la documentazione riserva <code>CellPaint</code> agli elementi visivi troppo complessi per l'HTML.

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 3. Puoi aprire tu stesso [module-03.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 VisualOperationsStudio 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 3 - "Disegno delle celle della DataGridView" - del corso LearnWisej "Disegno e grafica 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/drawing-and-painting/specs/module-03.md>

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

1. Progetto "VisualOperationsStudio", 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: Dai a VisualOperationsStudio una griglia di operations con due colonne disegnate a mano. Aggiungi colonne esplicite a gridOperations, imposta UserPaint = true solo sulle colonne Health e Trend e gestisci CellPaint una volta sola: esci immediatamente quando e.RowIndex è negativo o e.ColumnIndex non è una delle due colonne bersaglio, poi disegna una barra di avanzamento colorata per soglia per Health e una sparkline a polilinea per Trend dentro e.ClipRectangle. Leggi i valori di ogni riga dall'oggetto MachineStatus associato invece di cercare dentro una lista nell'handler, rispetta i colori della riga selezionata e tieni l'handler privo di effetti collaterali. Associa 1.000 righe generate, scorri la griglia e poi costruisci lo stesso indicatore di stato una seconda volta come cella AllowHtml, così da poter dichiarare per iscritto quale dei due manderesti in produzione.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - gridOperations con colonne esplicite e UserPaint = true solo sulle colonne Health e Trend
 - Un unico handler di CellPaint che verifica e.RowIndex e e.ColumnIndex prima di disegnare qualsiasi cosa
 - Cella di avanzamento colorata per soglia e cella sparkline a polilinea disegnate dentro e.ClipRectangle
 - Valori di riga letti dall'oggetto MachineStatus associato, senza ricerche, allocazioni o creazione di controlli per cella
 - Una colonna di stato AllowHtml costruita in parallelo, più una nota che confronta le due soluzioni su uno scorrimento di 1.000 righe
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 collega ti passa un esempio WinForms di disegno delle celle che inizia chiamando e.PaintBackground(...). Spiega che cosa ti danno invece gli argomenti dell'evento in Wisej.NET e che cosa deve ora decidere da solo l'handler.
- Il tuo helper per la sparkline ha una larghezza di tracciato fissa a 100 pixel e sul tuo

schermo sembra a posto. Descrivi due azioni dell'utente che lo rompono e che cosa dovrebbe leggere l'helper al suo posto.

– La colonna di stato HTML e la colonna Health disegnata mostrano entrambe una macchina in stato critico. Argomenta quale delle due manderesti in produzione per una griglia che il personale di supporto cerca e legge con uno screen reader, e che cosa perderesti in entrambi i casi.

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 collega ti passa un esempio WinForms di disegno delle celle che inizia chiamando `e.PaintBackground(...)`. Spiega che cosa ti danno invece gli argomenti dell'evento in Wisej.NET e che cosa deve ora decidere da solo l'handler.
- ✓ Il tuo helper per la sparkline ha una larghezza di tracciato fissa a 100 pixel e sul tuo schermo sembra a posto. Descrivi due azioni dell'utente che lo rompono e che cosa dovrebbe leggere l'helper al suo posto.
- ✓ La colonna di stato HTML e la colonna Health disegnata mostrano entrambe una macchina in stato critico. Argomenta quale delle due manderesti in produzione per una griglia che il personale di supporto cerca e legge con uno screen reader, e che cosa perderesti in entrambi i casi.

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.



Per questo corso non c'è ancora un esempio di riferimento — il tuo è il primo

Il [repository degli esempi LearnWisej](#) contiene build già fatte per quasi tutti i corsi, ma non ancora per questo. Significa che non c'è niente con cui confrontarsi e nessuna scorciatoia: la specifica e la tua revisione sono tutta la rete di sicurezza. Tieni quello che costruisci — e se viene bene, ci farebbe piacere vederlo.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito 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 VisualOperationsStudio 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 VisualOperationsStudio 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 VisualOperationsStudio 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.