

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.



Scrivere il codice è ormai la metà facile.

Ottenere codice funzionante per "Liste, alberi, repeater e dati gerarchici" da un assistente moderno richiede un buon prompt. Giudicarlo, correggerlo ed estenderlo richiede tutto quello che questo modulo ha trattato. È su questa divisione che è costruito l'esercizio: tu metti la specifica e la revisione, l'assistente mette la digitazione.

Che cosa stai chiedendo

Costruisci l'esploratore documenti di Operations Console: una TreeView di categorie o cartelle con un ID stabile associato a ogni nodo, una ListView di documenti legata al nodo selezionato, caricamento lazy o modalità virtuale su almeno un percorso di lista o albero, icone o image key per i diversi tipi di elemento, e uno UserControl di dettaglio che carica attraverso un servizio o un view-model ogni volta che la selezione cambia.

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

- ✓ TreeView di categorie con un ID stabile memorizzato su ogni nodo
- ✓ ListView di documenti legata al nodo selezionato nell'albero
- ✓ Caricamento lazy o modalità virtuale su almeno un percorso di lista o albero
- ✓ UserControl di dettaglio caricato tramite un servizio o un view-model alla selezione
- ✓ Icone o image key per i diversi tipi di elemento

Perché è fatto così

I controlli lista e albero vengono spesso trattati come semplici widget di selezione, ma in un'applicazione gestionale sono la mappa dell'utente: cartelle, clienti, ordini, procedure, ruoli, attività, documenti, cataloghi di prodotto e i moduli stessi dell'applicazione. L'esploratore documenti di Operations Console che costruisci in questo modulo deve restare veloce anche quando l'albero delle categorie è profondo e l'elenco dei documenti è lungo, e ogni clic deve corrispondere a un record di business reale, non a un pezzo di testo visualizzato.

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
forma dei dati	La prima domanda di progettazione: i dati sono piatti, raggruppati, gerarchici, ripetuti come card o un insieme di proprietà? La risposta sceglie il controllo.
viste della ListView	Un solo controllo che mostra la stessa collezione come LargeIcon, SmallIcon, Details o Tile: la scelta naturale per schermate in stile file manager o catalogo.
modalità virtuale	La modalità della ListView in cui l'applicazione imposta VirtualListSize e risponde a RetrieveVirtualItem, così gli elementi esistono solo per le righe che il controllo chiede.
caricamento lazy	Caricare all'avvio solo i livelli alti di un albero e recuperare i figli quando un nodo viene espanso, così l'avvio resta economico e l'interfaccia reattiva.
scorrimento virtuale e prefetch	Rendere nel browser solo le righe visibili di una lista o di un albero di grandi dimensioni, con PrefetchItems che ne prepara qualcuna in più per uno scorrimento fluido.
identità stabile	Un ID di business univoco memorizzato nel Tag o nel Name del nodo o dell'elemento, o in una mappa del view model, mai derivato dal testo visualizzato.
la selezione è un evento	Un cambio di selezione risolve un ID e chiede i dati a un servizio; nessuna query gira mai dentro il codice che costruisce nodi o elementi.

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 4. Puoi aprire tu stesso [module-04.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 4 - "Liste, alberi, repeater e dati gerarchici" - 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-04.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 l'esploratore documenti di Operations Console: una TreeView di categorie o cartelle con un ID stabile associato a ogni nodo, una ListView di documenti legata al nodo selezionato, caricamento lazy o modalità virtuale su almeno un percorso di lista o albero, icone o image key per i diversi tipi di elemento, e uno UserControl di dettaglio che carica attraverso un servizio o un view-model ogni volta che la selezione cambia.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - TreeView di categorie con un ID stabile memorizzato su ogni nodo
 - ListView di documenti legata al nodo selezionato nell'albero
 - Caricamento lazy o modalità virtuale su almeno un percorso di lista o albero
 - UserControl di dettaglio caricato tramite un servizio o un view-model alla selezione
 - Icone o image key per i diversi tipi di elemento
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:

- Due categorie sotto clienti diversi si chiamano entrambe "Contratti". Che cosa usa esattamente il tuo gestore di selezione per distinguerle, e dove vive quel valore?
- Se il numero di documenti sotto una categoria passasse da 300 a 300.000, quale parte del tuo esploratore si romperebbe per prima, e quale proprietà o evento cambieresti per sistemarla?
- Quali righe del tuo gestore di selezione potrebbero finire in uno unit test senza browser, e quali invece appartengono allo UserControl di dettaglio?

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:

- ✓ Due categorie sotto clienti diversi si chiamano entrambe "Contratti". Che cosa usa esattamente il tuo gestore di selezione per distinguerle, e dove vive quel valore?
- ✓ Se il numero di documenti sotto una categoria passasse da 300 a 300.000, quale parte del tuo esploratore si romperebbe per prima, e quale proprietà o evento cambieresti per sistemarla?
- ✓ Quali righe del tuo gestore di selezione potrebbero finire in uno unit test senza browser, e quali invece appartengono allo UserControl di dettaglio?

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 4 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 4 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Mastering the Control Library Course/Module 4`.



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 OperationsConsole 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 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."*



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.