

## 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.



### 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 OperationsConsole; 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

Sostituisci i segnaposto del Modulo 1 con lo shell vero di Operations Console: uno SplitContainer con la navigazione a sinistra, una superficie di dettaglio TabControl a destra, una ToolBar per i comandi frequenti e una StatusBar che mostra lo stato reale. Sposta l'interfaccia ripetuta dentro UserControl che espongono solo proprietà ed eventi pubblici significativi, realizza un layout per il profilo stretto con proprietà responsive o logica basata sul profilo, e documenta le tue scelte di docking e ancoraggio in LayoutNotes.md.

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

- ✓ Shell di navigazione costruito con SplitContainer, TabControl, ToolBar e StatusBar
- ✓ Interfaccia ripetuta spostata in UserControl con una superficie pubblica ridotta
- ✓ Layout per profilo stretto con proprietà responsive o logica basata sul profilo
- ✓ LayoutNotes.md che documenta le scelte di docking e ancoraggio
- ✓ Screenshot a larghezza desktop e ridotta

### Perché è fatto così

I contenitori sono l'architettura dell'interfaccia: decidono come i controlli vengono raggruppati, ridimensionati, navigati e riutilizzati. La Operations Console del Modulo 1 ha ancora pannelli segnaposto, e una schermata in cui ogni controllo è posizionato a mano diventa fragile al primo ridimensionamento del browser. I problemi di layout sono quasi sempre problemi di progettazione, non di pixel, e questo modulo sostituisce lo spostamento di pixel con un piano: regioni in dock, uno SplitContainer, un TabControl, un vero modello a comandi e UserControl che nascondono i propri interni.

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                                                                                                                                                                                                                      |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ordine di docking            | Il docking aggancia un figlio a un lato del contenitore o gli fa riempire quello che resta, e viene applicato nell'ordine dei figli: gli stessi quattro controlli producono shell diversi a seconda dell'ordine in cui sono stati aggiunti. |
| ancoraggio                   | L'ancoraggio lega i bordi di un controllo a quelli del contenitore; ancorato a sinistra e a destra si allarga con la larghezza, non ancorato su nessun asse resta "sospeso" mantenendo la posizione relativa.                               |
| SplitContainer               | Due regioni adiacenti che l'utente può ridimensionare: la scelta giusta ogni volta che deve essere la persona, e non lo sviluppatore, a decidere quanto spazio dare all'albero di navigazione o al pannello di dettaglio.                   |
| pannelli di layout           | FlowLayoutPanel dispone i figli uno accanto all'altro e va a capo, TableLayoutPanel allinea righe e colonne, e FlexLayoutPanel distribuisce e manda a capo come un layout responsive moderno.                                               |
| proprietà responsive         | Una proprietà che contiene un valore diverso per ciascun profilo client, impostata a design time: un layout desktop può così cambiare una manciata di proprietà sugli schermi stretti senza bisogno di una seconda schermata.               |
| modello a comandi            | Una regola coerente su dove sta un comando: MenuBar, ToolBar, RibbonBar, ContextMenu o StatusBar, ciascuno con il proprio compito, tutti che invocano comandi applicativi invece di contenere logica di business.                           |
| superficie dello UserControl | L'unità di riuso: un controllo composto che espone poche proprietà, eventi e metodi significativi invece di tutti i controlli figli, così che chi lo usa non si accoppi alla sua implementazione.                                           |

## 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 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 3 - "Contenitori, layout, navigazione e riuso" - 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-03.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: Sostituisci i segnaposto del Modulo 1 con lo shell vero di Operations Console: uno SplitContainer con la navigazione a sinistra, una superficie di dettaglio TabControl a destra, una ToolBar per i comandi frequenti e una StatusBar che mostra lo stato reale. Sposta l'interfaccia ripetuta dentro UserControl che espongono solo proprietà ed eventi pubblici significativi, realizza un layout per il profilo stretto con proprietà responsive o logica basata sul profilo, e documenta le tue scelte di docking e ancoraggio in LayoutNotes.md.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
  - Shell di navigazione costruito con SplitContainer, TabControl, ToolBar e StatusBar
  - Interfaccia ripetuta spostata in UserControl con una superficie pubblica ridotta
  - Layout per profilo stretto con proprietà responsive o logica basata sul profilo
  - LayoutNotes.md che documenta le scelte di docking e ancoraggio
  - Screenshot a larghezza desktop e ridotta
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 lo SplitContainer fosse aggiunto alla pagina prima della ToolBar e della StatusBar, che cosa vedrebbe l'utente e perché l'ordine conta?
- Quali membri del tuo UserControl potresti rinominare senza toccare nessuna delle pagine che lo usano, e quali no? Che cosa ti dice questo sulla sua superficie pubblica?
- Quali proprietà sono cambiate sul profilo stretto, e perché non hai costruito una schermata separata per gli schermi stretti?

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 lo SplitContainer fosse aggiunto alla pagina prima della ToolBar e della StatusBar, che cosa vedrebbe l'utente e perché l'ordine conta?
- ✓ Quali membri del tuo UserControl potresti rinominare senza toccare nessuna delle pagine che lo usano, e quali no? Che cosa ti dice questo sulla sua superficie pubblica?
- ✓ Quali proprietà sono cambiate sul profilo stretto, e perché non hai costruito una schermata separata per gli schermi stretti?

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 3 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 3 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Mastering the Control Library Course/Module 3`.



### 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.

- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/mastering-the-control-library/specs/module-03.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."*
- ✓ **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."*



### 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.