

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.



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

Crea lo shell del catalogo di controlli di Operations Console: una Page principale con un'area di navigazione, un'area contenuti, un'area comandi e un'area di stato; pagine segnaposto per Editors, Layouts, Lists and Trees, DataGridView, Dashboard e Widgets; una nota ControlSelection.md dentro il progetto che elenca quale famiglia di controlli userai per ciascuna area; nomi di controlli chiari ovunque; e un'area di stato che si aggiorna a ogni cambio di navigazione.

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

- ✓ Shell principale con aree di navigazione, contenuti, comandi e stato
- ✓ Pagine segnaposto per Editors, Layouts, Lists and Trees, DataGridView, Dashboard e Widgets
- ✓ ControlSelection.md con la famiglia di controlli scelta per ogni area
- ✓ Nomi di controlli chiari e un'area di stato che si aggiorna alla navigazione
- ✓ Screenshot e una breve nota che spiega le scelte sui controlli

Perché è fatto così

Wisej.NET include più di cento controlli e un team che non condivide un modello mentale su di essi finisce con venti schermate di ricerca diverse, cinque modi di segnalare un errore di validazione e JavaScript personalizzato dove sarebbe bastato un controllo nativo. Questo modulo fissa il vocabolario, le regole di scelta e le convenzioni su cui si appoggiano tutti i moduli successivi, e avvia lo shell di Operations Console.

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
albero dei controlli sul server	La gerarchia di oggetti controllo.NET che il server mantiene per ogni sessione; il tuo codice imposta proprietà e gestisce eventi qui, mai nel markup.
widget del browser	L'oggetto JavaScript che Wisej.NET crea per rappresentare un controllo lato server, tenuto allineato da cambi di stato serializzati, non da ricaricamenti di pagina.
famiglia di controlli	Un gruppo di controlli che condividono uno scopo (editor, contenitori, liste, griglie, contenuti, menu, notifiche): è qui che si prendono le decisioni.
extender	Un componente che aggiunge proprietà o comportamenti ai controlli esistenti a design time, senza doverli derivare.
Widget ed estensione	Il controllo Widget ospita un widget JavaScript personalizzato o di terze parti dentro il modello a eventi di Wisej.NET; un'estensione è un'integrazione già confezionata, di solito da NuGet.
UserControl	Una composizione riutilizzabile di controlli lato server con un significato di business e una superficie pubblica ridotta.
guida alla scelta dei controlli	Un documento vivo del team che fissa controlli standard, docking, stile di validazione e di notifica e convenzioni sulle griglie per ogni pattern di schermata.

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 1. Puoi aprire tu stesso [module-01.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 1 - "Modello mentale della libreria di controlli" - 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-01.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: Crea lo shell del catalogo di controlli di Operations Console: una Page principale con un'area di navigazione, un'area contenuti, un'area comandi e un'area di stato; pagine segnaposto per Editors, Layouts, Lists and Trees, DataGridView, Dashboard e Widgets; una nota ControlSelection.md dentro il progetto che elenca quale famiglia di controlli userai per ciascuna area; nomi di controlli chiari ovunque; e un'area di stato che si aggiorna a ogni cambio di navigazione.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Shell principale con aree di navigazione, contenuti, comandi e stato
 - Pagine segnaposto per Editors, Layouts, Lists and Trees, DataGridView, Dashboard e Widgets
 - ControlSelection.md con la famiglia di controlli scelta per ogni area
 - Nomi di controlli chiari e un'area di stato che si aggiorna alla navigazione
 - Screenshot e una breve nota che spiega le scelte sui controlli
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:

- Quando imposti statusLabel.Text nel metodo di navigazione, che cosa viaggia davvero verso il browser, e perché è diverso dallo scrivere HTML?
- Quale area dello shell trasformeresti per prima in uno UserControl, e quale sarebbe la sua superficie pubblica?
- Se un modulo successivo avesse bisogno di un'area di navigazione richiudibile, quale contenitore nativo sostituirebbe il pannello in dock, e perché è preferibile a un widget personalizzato?

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:

- ✓ Quando imposti `statusLabel.Text` nel metodo di navigazione, che cosa viaggia davvero verso il browser, e perché è diverso dallo scrivere HTML?
- ✓ Quale area dello shell trasformeresti per prima in uno `UserControl`, e quale sarebbe la sua superficie pubblica?
- ✓ Se un modulo successivo avesse bisogno di un'area di navigazione richiudibile, quale contenitore nativo sostituirebbe il pannello in dock, e perché è preferibile a un widget personalizzato?

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 1 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 1 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Mastering the Control Library Course/Module 1`.



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.

- ✓ **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."*
- ✓ **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-01.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."*



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.