

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.



L'assistente è la coppia di mani veloci. L'architetto resti tu.

Un assistente può produrre il progetto di questo modulo in pochi minuti. Quello che non può fare è decidere se il risultato è il progetto giusto per la tua applicazione: quella scelta resta tua. Il giudizio lo metti tu: istruiscilo per bene, poi verifica quello che ti torna indietro rispetto a ciò che il Modulo 7 ti ha appena insegnato.

Che cosa stai chiedendo

Incapsula un piccolo widget JavaScript di rating o di stato con il controllo Widget, oppure configura un pacchetto di estensione Wisej.NET: carica i suoi package JavaScript e CSS, solleva un evento lato server dal client con il pattern degli eventi widget, chiama una funzione lato client dal server con CallAsync o Eval, applica il tema della console, poi rifinisci il capstone completo di Operations Console e prepara una demo di 5 minuti.

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

- ✓ Schermata basata su Widget o estensione con i suoi package JavaScript e CSS caricati
- ✓ Evento client-verso-server sollevato con il pattern degli eventi widget o con un evento nativo dell'estensione
- ✓ Chiamata server-verso-client tramite CallAsync, Eval o l'API dell'estensione
- ✓ Contratto client/server documentato: package, proprietà, eventi e formato del payload
- ✓ Capstone di Operations Console rifinito con uno script di demo da 5 minuti

Perché è fatto così

Prima o poi un prodotto ha bisogno di qualcosa che la libreria nativa non offre: una capacità specializzata del browser, una libreria JavaScript di terze parti, una visualizzazione su misura o un'integrazione riutilizzabile. La tentazione è buttarsi nel JavaScript e perdere il modello di Wisej.NET: stato sul server, temi, designer e un flusso di eventi che un altro sviluppatore riesce a seguire. Questo modulo integra un widget client senza rinunciare a nulla di tutto ciò, e poi porta a termine il capstone 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
controllo nativo	Un controllo che implementa sia il lato server sia il proprio widget nel browser, quindi ha già supporto a design time, temi, eventi e stato sul server; è sempre la prima tappa.
tipi di estensione	Control, Widget, Component, Extender Provider e Icon Pack: dare un nome al tipo evita di confondere "riutilizzabile" con "nuovo controllo nativo".
controllo Widget	Un contenitore client generico che carica package JavaScript e CSS, esegue uno script di inizializzazione e scambia eventi e dati con il codice sul server.
Packages e InitScript	L'elenco dei package dichiara i file client da cui il widget dipende; l'InitScript crea l'oggetto JavaScript, collega i suoi eventi client ed espone solo le funzioni che servono al server.
WidgetEvent	L'evento server sollevato quando il codice client chiama fireWidgetEvent; il suo tipo e il payload arrivano dal browser e vanno validati come qualsiasi altro input dell'utente.
CallAsync / Eval	La metà server-verso-client del collegamento: chiama una funzione esposta dal widget, oppure valuta uno script, senza uscire da C#.
contratto del widget	Le proprietà, gli eventi, il formato del payload, le dipendenze dai package e le regole di tema documentati, che permettono a un altro sviluppatore — o a un assistente IA ben istruito — di mantenere il widget.

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 7. Puoi aprire tu stesso [module-07.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 7 - "Widget personalizzati, estensioni, temi e capstone" - 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-07.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: Incapsula un piccolo widget JavaScript di rating o di stato con il controllo Widget, oppure configura un pacchetto di estensione Wisej.NET: carica i suoi package JavaScript e CSS, solleva un evento lato server dal client con il pattern degli eventi widget, chiama una funzione lato client dal server con `CallAsync` o `Eval`, applica il tema della console, poi rifinisci il capstone completo di Operations Console e prepara una demo di 5 minuti.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Schermata basata su Widget o estensione con i suoi package JavaScript e CSS caricati
 - Evento client-verso-server sollevato con il pattern degli eventi widget o con un evento nativo dell'estensione
 - Chiamata server-verso-client tramite `CallAsync`, `Eval` o l'API dell'estensione
 - Contratto client/server documentato: package, proprietà, eventi e formato del payload
 - Capstone di Operations Console rifinito con uno script di demo da 5 minuti
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:

- Che cosa succederebbe se un utente modificasse il payload di `ratingChanged` negli strumenti per sviluppatori del browser prima che arrivi al tuo gestore, e quale riga del tuo codice lo blocca?
- Il tuo widget si mostra vuoto dopo che il tab Widgets è stato nascosto e riaperto: quali tre cose controlli, in quale ordine, e perché l'HTML di avvio non è la soluzione?
- Quale regola del tuo widget potresti spostare in JavaScript come ottimizzazione, e quale non deve mai lasciare il server?

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:

- ✓ Che cosa succederebbe se un utente modificasse il payload di `ratingChanged` negli strumenti per sviluppatori del browser prima che arrivi al tuo gestore, e quale riga del tuo codice lo blocca?
- ✓ Il tuo widget si mostra vuoto dopo che il tab Widgets è stato nascosto e riaperto: quali tre cose controlli, in quale ordine, e perché l'HTML di avvio non è la soluzione?
- ✓ Quale regola del tuo widget potresti spostare in JavaScript come ottimizzazione, e quale non deve mai lasciare il server?

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 7 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 7 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Mastering the Control Library Course/Module 7`.



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.

- ✓ **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-07.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."*



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.