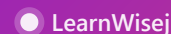


Esercizio di coding con l'IA

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 2 ti ha appena insegnato.

Che cosa stai chiedendo

Costruisci uno UserControl CustomerEditor per la Operations Console con campi etichettati per nome, email, stato, data di inizio, fido e tipo cliente, usando per ogni valore l'editor adatto invece di una TextBox per tutto. Aggiungi messaggi ErrorProvider per i campi obbligatori, la forma dell'email, l'intervallo del fido e le date non valide; comandi Save, Reset e Validate con comportamento disabilitato e di attesa durante il salvataggio; e un Toast o un MessageBox quando la validazione o il salvataggio si conclude.

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

- ✓ UserControl CustomerEditor con campi etichettati per nome, email, stato, data di inizio, fido e tipo cliente
- ✓ Editor coerenti con il valore: TextBox, DateTimePicker, NumericUpDown e ComboBox invece di testo libero ovunque
- ✓ Messaggi ErrorProvider per campi obbligatori, forma dell'email, intervallo del fido e date non valide
- ✓ Comandi Save, Reset e Validate con comportamento disabilitato e di attesa durante il salvataggio
- ✓ Feedback con Toast o MessageBox quando la validazione o il salvataggio si conclude

Perché è fatto così

Gli editor sono il primo punto in cui una libreria di controlli diventa un sistema gestionale. Una maschera che sembra corretta ma accetta input ambigui o non validi genera ticket di assistenza e script di bonifica molto tempo dopo il rilascio. Gli editor di Wisej.NET seguono il familiare modello a componenti di .NET ma girano nel browser, quindi una stessa regola di business può essere applicata su più livelli: proprietà degli editor, extender di validazione, gestori di eventi, data binding, validazione nei servizi e vincoli del database.

Il compito del livello UI è impedire gli errori evidenti e spiegare chiaramente quelli che restano; non deve mai essere l'unico livello che protegge l'applicazione.

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
editor coerente con il valore	L'editor il cui tipo vincola già l'input: DateTimePicker per le date, NumericUpDown per i numeri con limiti, ComboBox per i valori da elenco, così il server non deve mai interpretare testo ambiguo.
modello a comandi	Una distinzione voluta tra comandi primari (Save, Approve, Search, Send, Export), comandi secondari (ToolBar, MenuBar, SplitButton, menu contestuale) e comandi distruttivi, che richiedono conferma o una collocazione prudente.
ErrorProvider	L'extender che associa un messaggio specifico e utile al controllo che è sbagliato, invece di un generico "Valore non valido" da qualche altra parte nella pagina.
feedback bloccante e non bloccante	MessageBox interrompe per una decisione o una conferma critica; AlertBox e Toast comunicano lo stato senza fermare il flusso di lavoro.
validatori con un nome	Metodi come ValidateEmail o ValidateCreditLimit che contengono una regola ciascuno, così la regola è testabile e spiegabile invece di essere sepolta in un gestore del click.
stato di attesa	Disabilitare Save e mostrare un avanzamento mentre un'operazione è in corso: evita gli invii doppi ed è parte dell'affidabilità, non decorazione.
UserControl editor	Una composizione di label, editor, un ErrorProvider e un metodo di validazione che restituisce un oggetto risultato, così che il resto dell'applicazione non sappia mai come viene controllato il campo email.

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 2. Puoi aprire tu stesso [module-02.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 2 - "Editor, pulsanti, validazione e feedback" - 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-02.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 uno UserControl CustomerEditor per la Operations Console con campi etichettati per nome, email, stato, data di inizio, fido e tipo cliente, usando per ogni valore l'editor adatto invece di una TextBox per tutto. Aggiungi messaggi ErrorProvider per i campi obbligatori, la forma dell'email, l'intervallo del fido e le date non valide; comandi Save, Reset e Validate con comportamento disabilitato e di attesa durante il salvataggio; e un Toast o un MessageBox quando la validazione o il salvataggio si conclude.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - UserControl CustomerEditor con campi etichettati per nome, email, stato, data di inizio, fido e tipo cliente
 - Editor coerenti con il valore: TextBox, DateTimePicker, NumericUpDown e ComboBox invece di testo libero ovunque
 - Messaggi ErrorProvider per campi obbligatori, forma dell'email, intervallo del fido e date non valide
 - Comandi Save, Reset e Validate con comportamento disabilitato e di attesa durante il salvataggio
 - Feedback con Toast o MessageBox quando la validazione o il salvataggio si conclude
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:

- Quali errori di validazione non possono più verificarsi grazie all'editor che hai scelto, e quali hanno ancora bisogno di un messaggio ErrorProvider?
- Se l'utente fa clic su Save due volte in un secondo, che cosa fa la tua schermata, e in quale punto del codice è garantito?
- Che cosa sa la pagina shell su come viene validato il campo email, e che cosa dovrebbe cambiare se quella regola si spostasse in un servizio?

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:

- ✓ Quali errori di validazione non possono più verificarsi grazie all'editor che hai scelto, e quali hanno ancora bisogno di un messaggio ErrorProvider?
- ✓ Se l'utente fa clic su Save due volte in un secondo, che cosa fa la tua schermata, e in quale punto del codice è garantito?
- ✓ Che cosa sa la pagina shell su come viene validato il campo email, e che cosa dovrebbe cambiare se quella regola si spostasse in un servizio?

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 2 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 2 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Mastering the Control Library Course/Module 2`.



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-02.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.