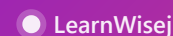


Esercizio di coding con l'IA

Costruisci il progetto SupportDesk 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.



Lo stesso modulo, costruito come costruirai davvero le cose.

Tutte le altre sezioni di questo modulo spiegano come funziona "Validazione, ErrorProvider e feedback all'utente". Questa ti chiede di produrlo: dirigendo un assistente con la specifica del modulo stesso, poi verificando che il risultato giri e faccia quello che descrive la traccia del lab. È il flusso di lavoro che userai sul lavoro vero, provato su qualcosa di cui conosci già la risposta giusta.

Che cosa stai chiedendo

Aggiungi la validazione all'editor dei ticket di Support Desk: DataAnnotations su TicketEditModel, un TicketValidator che restituisce errori per campo e di riepilogo, un ErrorProvider azzerato prima di ogni esecuzione e impostato per Title, Customer, Category e DueDate, una label di riepilogo per le regole tra più campi, Save disabilitato finché il modello non è valido, un gestore di DbUpdateException che mostra un messaggio comprensibile per un numero di ticket duplicato, e almeno cinque casi di test negativi.

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

- ✓ DataAnnotations su TicketEditModel e un TicketValidator che restituisce errori per campo e di riepilogo
- ✓ ErrorProvider azzerato prima di ogni esecuzione e impostato per Title, Customer, Category e DueDate
- ✓ Label di riepilogo per le regole tra più campi e Save disabilitato finché il modello non è valido
- ✓ DbUpdateException gestita con un messaggio comprensibile per un numero di ticket duplicato
- ✓ Almeno cinque casi di test negativi

Perché è fatto così

L'editor dei ticket del Modulo 4 salva qualsiasi cosa l'utente abbia digitato. Un titolo vuoto, un ticket senza cliente o un ticket chiuso con scadenza la settimana prossima passano tutti per SaveChangesAsync e vengono fermati, se va bene, dal database, che risponde con un'eccezione del provider che un agente di supporto non sa leggere. Gli utenti hanno bisogno di messaggi immediati e comprensibili prima che il database rifiuti un record, e l'applicazione ha bisogno di un'ultima linea di difesa che resti al suo posto quando l'interfaccia viene aggirata da un'importazione, da una seconda schermata o da un bug.

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
tre livelli di validazione	La validazione dell'interfaccia dà un feedback immediato, la validazione di dominio impone regole che devono valere ovunque e i vincoli del database garantiscono la coerenza finale; ogni livello intercetta ciò che è sfuggito a quello sopra.
DataAnnotations sul modello di modifica	Attributi come [Required] e [StringLength] su TicketEditModel descrivono le regole di input dell'utente in un unico punto dichiarativo, separato dall'entità EF Core.
validazione esplicita	Validator.TryValidateObject esegue le annotation su richiesta; EF Core non le esegue per te prima di SaveChangesAsync, quindi è il form a doverlo chiamare.
TicketValidator e ValidationMessage	Un servizio che restituisce elementi ValidationMessage(FieldName, Message), così l'insieme di regole è testabile e il form si limita a mappare i messaggi sui controlli.
ErrorProvider	L'extender Wisej.NET che mostra un'icona e un tooltip accanto al controllo non valido tramite SetError, e che può anche essere collegato agli errori di dati dell'elemento corrente.
riepilogo di validazione	Una label o un MessageBox che elenca gli errori tra più campi che non appartengono a un singolo controllo, come un ticket chiuso con una data di scadenza futura.
DbUpdateException	L'eccezione che EF Core solleva quando il database rifiuta la modifica; è l'ultimo livello di sicurezza e va tradotta in un messaggio comprensibile, mai mostrata così com'è.

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 5. Puoi aprire tu stesso [module-05.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 SupportDesk 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 5 - "Validazione, ErrorProvider e feedback all'utente" - del corso LearnWisej "Data binding con EF Core", 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/ef-core-binding/specs/module-05.md>

Poi costruisci un progetto Wisej.NET 4 completo che io possa aprire in Visual Studio:

1. Progetto "SupportDesk", 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: Aggiungi la validazione all'editor dei ticket di Support Desk: DataAnnotations su `TicketEditModel`, un `TicketValidator` che restituisce errori per campo e di riepilogo, un `ErrorProvider` azzerato prima di ogni esecuzione e impostato per Title, Customer, Category e DueDate, una label di riepilogo per le regole tra più campi, Save disabilitato finché il modello non è valido, un gestore di `DbUpdateException` che mostra un messaggio comprensibile per un numero di ticket duplicato, e almeno cinque casi di test negativi.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - DataAnnotations su `TicketEditModel` e un `TicketValidator` che restituisce errori per campo e di riepilogo
 - `ErrorProvider` azzerato prima di ogni esecuzione e impostato per Title, Customer, Category e DueDate
 - Label di riepilogo per le regole tra più campi e Save disabilitato finché il modello non è valido
 - `DbUpdateException` gestita con un messaggio comprensibile per un numero di ticket duplicato
 - Almeno cinque casi di test negativi
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 togliessi tutte le DataAnnotations da `TicketEditModel` ma le lasciassi sull'entità Ticket, quali dei tuoi cinque test negativi fallirebbero ancora prima di `SaveChangesAsync`, e perché?
- La regola sui ticket chiusi riporta DueDate come campo. Che cosa cambierebbe per l'utente se non riportasse alcun campo, e quando sarebbe la scelta migliore?
- Due agenti creano un ticket nello stesso secondo e il secondo salvataggio lancia `DbUpdateException`. Che cosa vede il secondo agente, che cosa registra il server, e quale dei due mostreresti a un cliente?

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 togliessi tutte le DataAnnotations da TicketEditModel ma le lasciassi sull'entità Ticket, quali dei tuoi cinque test negativi fallirebbero ancora prima di SaveChangesAsync, e perché?
- ✓ La regola sui ticket chiusi riporta DueDate come campo. Che cosa cambierebbe per l'utente se non riportasse alcun campo, e quando sarebbe la scelta migliore?
- ✓ Due agenti creano un ticket nello stesso secondo e il secondo salvataggio lancia DbUpdateException. Che cosa vede il secondo agente, che cosa registra il server, e quale dei due mostreresti a un cliente?

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 5 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 5 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Data Binding with EF Core Course/Module 5`.



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 SupportDesk 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 SupportDesk 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/ef-core-binding/specs/module-05.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.