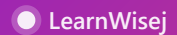


Esercizio di coding con l'IA

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.



Scrivere il codice è ormai la metà facile.

Ottenere codice funzionante per "Binding bidirezionale ed editor CRUD" da un assistente moderno richiede un buon prompt. Giudicarlo, correggerlo ed estenderlo richiede tutto quello che questo modulo ha trattato. È su questa divisione che è costruito l'esercizio: tu metti la specifica e la revisione, l'assistente mette la digitazione.

Che cosa stai chiedendo

Costruisci i form modali Add Ticket ed Edit Ticket per la Support Desk Data Console: un TicketEditModel, un TicketEditorForm con una BindingSource e un ErrorProvider, proprietà di TextBox, ComboBox, DateTimePicker e CheckBox collegate al modello, LoadEditorAsync per ticket nuovi ed esistenti, SaveAsync che chiama EndEdit, un segnaposto per la validazione, la mappatura e SaveChangesAsync, Delete con conferma e un caricamento aggiornato dell'entità, e una griglia padre che si aggiorna dopo DialogResult.OK.

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

- ✓ TicketEditModel e TicketEditorForm con una BindingSource e un ErrorProvider
- ✓ Proprietà di TextBox, ComboBox, DateTimePicker e CheckBox collegate al modello di modifica
- ✓ LoadEditorAsync per ticket nuovi ed esistenti e SaveAsync con EndEdit, mappatura e SaveChangesAsync
- ✓ Delete con conferma, un caricamento aggiornato dell'entità e una gestione pulita dei record già eliminati
- ✓ Griglia padre aggiornata dopo DialogResult.OK, e Cancel che non salva nulla

Perché è fatto così

Gli operatori di Support Desk vogliono schermate Add Ticket ed Edit Ticket che si comportino come form desktop: un dialogo modale, i pulsanti Save, Cancel e Delete, e una griglia che mostra la modifica nell'istante in cui il dialogo si chiude. Il binding bidirezionale rende quella schermata rapida da costruire, ma Save è un'operazione deliberata, non un effetto collaterale della digitazione. Se l'editor legge un valore collegato troppo presto, salva una riga della griglia senza tracking come se fosse tracciata, o lascia che un doppio click invii due volte, il form sembra comunque corretto mentre perde o duplica dati in silenzio. Questo modulo rende espliciti salvataggio, annullamento, eliminazione e ricaricamento, così l'editor dà la sensazione di un form desktop e segue comunque regole di accesso ai dati sicure per il web.

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
binding bidirezionale	Una proprietà di un controllo come TextBox.Text è collegata a una proprietà dell'elemento corrente, così il controllo mostra il modello e il modello riceve ciò che l'utente ha digitato.
DataSourceUpdateMode	L'argomento di DataBindings.Add che decide quando un controllo spinge il suo valore nel modello: alla validazione (quando il focus esce) oppure a ogni cambio di proprietà.
EndEdit e CancelEdit	BindingSource.EndEdit conferma nell'elemento corrente le modifiche in sospeso dei controlli; CancelEdit le scarta. Save inizia sempre con EndEdit.
AddNew e RemoveCurrent	Operazioni di lista sulla BindingSource che creano un nuovo elemento corrente o rimuovono quello selezionato dalla lista collegata; nessuna delle due tocca il database finché non salvi.
binding diretto alle entità	Collegare i controlli direttamente a un'entità EF Core tracciata, così il change tracker sa già che cosa è cambiato. Produttivo per form piccoli, di breve durata e con un Context privato.
modello di modifica	Una classe solo per l'interfaccia come TicketEditModel che viene collegata, validata e poi mappata sull'entità, dando a validazione, formattazione e filtri di sicurezza un confine tutto loro.
salvataggio ed eliminazione con Context nuovo	Ogni Save e ogni Delete crea il proprio DbContext dalla factory, carica l'entità per chiave, la modifica e chiama SaveChangesAsync, così la fonte di verità è il database, non la riga della griglia.

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 4. Puoi aprire tu stesso [module-04.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 4 - "Binding bidirezionale ed editor CRUD" - 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-04.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: Costruisci i form modali Add Ticket ed Edit Ticket per la Support Desk Data Console: un `TicketEditModel`, un `TicketEditorForm` con una `BindingSource` e un `ErrorProvider`, proprietà di `TextBox`, `ComboBox`, `DateTimePicker` e `CheckBox` collegate al modello, `LoadEditorAsync` per ticket nuovi ed esistenti, `SaveAsync` che chiama `EndEdit`, un segnaposto per la validazione, la mappatura e `SaveChangesAsync`, `Delete` con conferma e un caricamento aggiornato dell'entità, e una griglia padre che si aggiorna dopo `DialogResult.OK`.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - `TicketEditModel` e `TicketEditorForm` con una `BindingSource` e un `ErrorProvider`
 - Proprietà di `TextBox`, `ComboBox`, `DateTimePicker` e `CheckBox` collegate al modello di modifica
 - `LoadEditorAsync` per ticket nuovi ed esistenti e `SaveAsync` con `EndEdit`, mappatura e `SaveChangesAsync`
 - `Delete` con conferma, un caricamento aggiornato dell'entità e una gestione pulita dei record già eliminati
 - Griglia padre aggiornata dopo `DialogResult.OK`, e `Cancel` che non salva nulla
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:

- Quale ciclo di vita hai scelto per il `DbContext` nell'editor, e che cosa dovrebbe essere vero del form perché un `Context` legato al form diventi accettabile?
- Quali valori sono collegati all'interfaccia, quali vivono solo nel modello di modifica e quali restano solo nel database, e perché `Number` non è modificabile?
- Quali modalità di errore hai gestito esplicitamente (doppio click, ticket già eliminato, rifiuto del database, modifica concorrente), e che cosa vede l'operatore in ciascun caso?

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:

- ✓ Quale ciclo di vita hai scelto per il DbContext nell'editor, e che cosa dovrebbe essere vero del form perché un Context legato al form diventi accettabile?
- ✓ Quali valori sono collegati all'interfaccia, quali vivono solo nel modello di modifica e quali restano solo nel database, e perché Number non è modificabile?
- ✓ Quali modalità di errore hai gestito esplicitamente (doppio click, ticket già eliminato, rifiuto del database, modifica concorrente), e che cosa vede l'operatore in ciascun caso?

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 4 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 4 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Data Binding with EF Core Course/Module 4`.



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.

- ✓ **Sostituisci un pezzo senza spostare le giunzioni.** *"Scegli la parte di questo progetto SupportDesk 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."*
- ✓ **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."*



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.