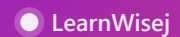


Esercizio di coding con l'IA

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.



Istruiscilo, leggilo, poi mettilo alla prova.

Questo esercizio ha tre parti, e solo la prima riguarda il prompt. Istruisci un assistente con la specifica del modulo e ti fai restituire una soluzione SupportDesk; leggi quello che ha scritto e decidi se fa davvero ciò che il modulo chiedeva; poi lo spingi più in là di quanto il modulo chiedesse. La seconda e la terza parte sono dove si impara.

Che cosa stai chiedendo

Costruisci il browser dei ticket di Support Desk: i record `TicketListItem` e `TicketSearchCriteria`, un metodo `SearchTicketsAsync` che compone `AsNoTracking`, `Where`, `OrderBy`, `Skip`, `Take` e `Select` prima di `ToListAsync`, una `DataGridView` collegata tramite una `BindingSource` a `TicketListItem`, `ComboBox` di lookup per stato e cliente caricate prima della prima ricerca, i pulsanti `Search`, `Next Page` e `Previous Page`, la visualizzazione del conteggio totale e della dimensione di pagina, e una protezione di caricamento che mantiene l'interfaccia utilizzabile dopo un errore.

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

- ✓ Record `TicketListItem` e `TicketSearchCriteria` con `SearchTicketsAsync`
- ✓ `DataGridView` collegata a `TicketListItem` tramite una `BindingSource`
- ✓ Filtri e paginazione composti nella query così che il lavoro lo faccia il database
- ✓ `ComboBox` di lookup caricate prima della prima ricerca, che mostrano i nomi e memorizzano le chiavi
- ✓ Pulsanti `Search`, `Next Page` e `Previous Page` con un conteggio totale e una protezione di caricamento

Perché è fatto così

Gli operatori di Support Desk hanno bisogno di una griglia dei ticket che cerchi rapidamente senza trascinare l'intero database nella memoria del server. In un'applicazione Wisej.NET ogni sessione tiene sul server i propri form, le proprie `BindingSource` e le proprie righe selezionate, quindi una schermata che carica migliaia di entità `Ticket` tracciate per utente moltiplica quel costo per il numero di browser aperti. Questo modulo costruisce il browser dei ticket nel modo giusto: componi la query, eseguila in modo asincrono, materializza una piccola pagina proiettata e passa quella lista all'interfaccia tramite una `BindingSource`.

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
metodo di esecuzione asincrono	ToListAsync, CountAsync e SingleOrDefaultAsync sono le chiamate che inviano davvero l'SQL; attenderle con await lascia libero il thread del server mentre il database lavora.
proiezione	Una Select in un record TicketListItem restituisce esattamente le colonne che la griglia mostra, invece dell'entità Ticket con il suo grafo di Customer, Agent e Comments.
query senza tracking	AsNoTracking dice al Context di non fare lo snapshot dei risultati: meno memoria, nessun aggiornamento accidentale e una lista indipendente da qualsiasi unit of work di modifica.
BindingSource	Il componente Wisej.NET tra una collezione di dati e i controlli collegati; gestisce l'elemento corrente, la navigazione e le operazioni sulla lista e mantiene sincronizzati la griglia e i controlli di dettaglio.
IQueryable	Una ricetta per una query sul database, non una lista stabile: deve essere eseguita e materializzata prima di assegnare qualsiasi cosa all'interfaccia.
protezione di caricamento	Un flag, pulsanti disabilitati e una label di stato attorno a ogni chiamata al database attesa con await, ripristinati nel finally, così gira un'operazione alla volta e la pagina resta utilizzabile dopo un errore.
binding dei lookup	Una ComboBox con DisplayMember per il nome mostrato e ValueMember per la chiave memorizzata, caricata prima che parta la prima ricerca o che venga assegnato il modello di modifica.

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 3. Puoi aprire tu stesso [module-03.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 3 - "Caricare i dati in BindingSource, DataGridView e controlli di lookup" - 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-03.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 il browser dei ticket di Support Desk: i record `TicketListItem` e `TicketSearchCriteria`, un metodo `SearchTicketsAsync` che compone `AsNoTracking`, `Where`, `OrderBy`, `Skip`, `Take` e `Select` prima di `ToListAsync`, una `DataGridView` collegata tramite una `BindingSource` a `TicketListItem`, `ComboBox` di lookup per stato e cliente caricate prima della prima ricerca, i pulsanti `Search`, `Next Page` e `Previous Page`, la visualizzazione del conteggio totale e della dimensione di pagina, e una protezione di caricamento che mantiene l'interfaccia utilizzabile dopo un errore.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Record `TicketListItem` e `TicketSearchCriteria` con `SearchTicketsAsync`
 - `DataGridView` collegata a `TicketListItem` tramite una `BindingSource`
 - Filtri e paginazione composti nella query così che il lavoro lo faccia il database
 - `ComboBox` di lookup caricate prima della prima ricerca, che mostrano i nomi e memorizzano le chiavi
 - Pulsanti `Search`, `Next Page` e `Previous Page` con un conteggio totale e una protezione di caricamento
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` dentro `SearchTicketsAsync`, e che cosa si romperebbe se la pagina tenesse vivo un solo `Context` per tutta la sua durata?
- Dopo una ricerca, quali dati vengono tenuti sul server per questa sessione e quali restano solo nel database? Quanto costa un secondo operatore in contemporanea?
- Se il lookup dei clienti crescesse fino a cinquantamila righe, quali parti della pagina cambieresti e quali parti del servizio di query resterebbero esattamente come sono?

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 dentro SearchTicketsAsync, e che cosa si romperebbe se la pagina tenesse vivo un solo Context per tutta la sua durata?
- ✓ Dopo una ricerca, quali dati vengono tenuti sul server per questa sessione e quali restano solo nel database? Quanto costa un secondo operatore in contemporanea?
- ✓ Se il lookup dei clienti crescesse fino a cinquantamila righe, quali parti della pagina cambieresti e quali parti del servizio di query resterebbero esattamente come sono?

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 3 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 3 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Data Binding with EF Core Course/Module 3`.



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.

- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice SupportDesk 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."*
- ✓ **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."*



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.