

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.



Questo progetto non lo scriverai a mano. Lo dirigerai.

Quasi nessuno scrive più codice Wisej.NET da un file vuoto: un assistente scrive la prima bozza e lo sviluppatore decide se è giusta. Funziona però solo se conosci la materia, ed è quello che ti ha dato il resto di questo modulo. Qui affidi quelle conoscenze a ChatGPT o a Claude, ti fai restituire il progetto del modulo come soluzione Visual Studio funzionante, e lo rivedi come rivedresti la pull request di un collega.

Che cosa stai chiedendo

Crea la soluzione Support Desk Data Console con i progetti Web, Data, Services e Tests, aggiungi i pacchetti del provider EF Core al progetto dati, crea SupportDeskContext con una factory di design time, registra AddDbContextFactory nello startup ASP.NET Core ed esponi a Wisej.NET IServiceProvider di Microsoft, poi risolvi un servizio di query da una Page Wisej.NET ed esegui una prima query di conteggio asincrona protetta da un flag di caricamento e da un messaggio di errore, senza ancora collegare nulla.

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

- ✓ Soluzione con i progetti Web, Data, Services e Tests e i pacchetti del provider EF Core
- ✓ SupportDeskContext registrato tramite AddDbContextFactory con una factory di design time
- ✓ IServiceProvider di Microsoft esposto a Wisej.NET e un servizio di query risolto da una Page
- ✓ Prima query di conteggio asincrona con una protezione di caricamento e un messaggio di errore
- ✓ Nessun DbContext statico in tutta la soluzione

Perché è fatto così

Un'applicazione Wisej.NET sembra WinForms perché form, pagine, controlli, componenti BindingSource e gestori di eventi vivono tutti sul server, e questa familiarità invita a una scorciatoia pericolosa: trattare un DbContext come un altro oggetto server di lunga durata. Anche EF Core gira sul server, ma non è un contenitore di stato dell'interfaccia; è una breve unit of work il cui change tracker non è thread-safe né pensato per essere condiviso. Molte sessioni del browser apriranno la Support Desk Data Console contemporaneamente, quindi la prima competenza di questo corso è tenere separati i cicli di vita dell'interfaccia e quello del database, prima ancora di collegare qualsiasi griglia.

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
stato di sessione	Lo stato lato server per ogni sessione del browser che Wisej.NET mantiene vivo: pagine aperte, riga selezionata, filtri, messaggi di validazione e modifiche in sospeso, raggiungibile tramite Application.Session e l'albero dei controlli attivo.
ciclo di vita degli oggetti dell'interfaccia	Quanto esiste una Page, un Form o una BindingSource, dalla costruzione finché l'utente non la chiude o la sessione termina; spesso ore, molto più di qualsiasi chiamata al database.
ciclo di vita di richiesta/thread	Un evento del browser o una continuazione di await sul server; il thread che ha avviato un gestore di click non è necessariamente quello che lo termina.
unit of work	Il ciclo di vita di un DbContext: crealo, interroga o modifica le entità, chiama SaveChangesAsync, eliminalo; secondi, non sessioni.
DbContextFactory	Il servizio IDbContextFactory<T> registrato da AddDbContextFactory che fornisce un Context nuovo e configurato correttamente ogni volta che un'operazione ne ha bisogno.
ponte verso la DI di Microsoft	La registrazione dell' IServiceProvider di ASP.NET Core in Application.Services, così le pagine Wisej.NET possono risolvere gli stessi servizi configurati allo startup.
fuga di dati tra sessioni	Un oggetto mutabile statico o condiviso (un Context, un change tracker, una lista collegata, un'entità selezionata) che la sessione di un utente può leggere dopo che la sessione di un altro utente l'ha scritto.

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 1. Puoi aprire tu stesso [module-01.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 1 - "Architettura: EF Core dentro un'applicazione Wisej.NET" - 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-01.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: Crea la soluzione Support Desk Data Console con i progetti Web, Data, Services e Tests, aggiungi i pacchetti del provider EF Core al progetto dati, crea `SupportDeskContext` con una factory di design time, registra `AddDbContextFactory` nello startup ASP.NET Core ed esponi a Wisej.NET l'`IServiceProvider` di Microsoft, poi risolvi un servizio di query da una Page Wisej.NET ed esegui una prima query di conteggio asincrona protetta da un flag di caricamento e da un messaggio di errore, senza ancora collegare nulla.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Soluzione con i progetti Web, Data, Services e Tests e i pacchetti del provider EF Core
 - `SupportDeskContext` registrato tramite `AddDbContextFactory` con una factory di design time
 - `IServiceProvider` di Microsoft esposto a Wisej.NET e un servizio di query risolto da una Page
 - Prima query di conteggio asincrona con una protezione di caricamento e un messaggio di errore
 - Nessun `DbContext` statico in tutta la soluzione
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` dietro `CountTicketsAsync`, e che cosa andrebbe storto se la pagina tenesse vivo quel Context tra un click e l'altro?
- Due utenti aprono la console Support Desk nello stesso momento ed entrambi cliccano `Count`. Quali oggetti sono separati per sessione, quali sono condivisi, e perché questa divisione è sicura?
- Se il database non è raggiungibile, che cosa vede l'utente, che cosa fa il pulsante, e come torna a false il flag di caricamento?

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 dietro CountTicketsAsync, e che cosa andrebbe storto se la pagina tenesse vivo quel Context tra un click e l'altro?
- ✓ Due utenti aprono la console Support Desk nello stesso momento ed entrambi cliccano Count. Quali oggetti sono separati per sessione, quali sono condivisi, e perché questa divisione è sicura?
- ✓ Se il database non è raggiungibile, che cosa vede l'utente, che cosa fa il pulsante, e come torna a false il flag di caricamento?

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 1 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 1 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Data Binding with EF Core Course/Module 1`.



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 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-01.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 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."*



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.