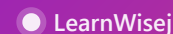


Esercizio di coding con l'IA

Costruisci il progetto EnterpriseOps 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 "Architettura dei dati reale: EF Core, transazioni, repository e comandi" 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

Sostituisci la persistenza fittizia di WorkOrderService con un'implementazione basata su EF Core o con un'astrazione repository adatta alla produzione. Implementa le operazioni Create, Update, Approve, Search e Audit con confini transazionali ed errori mappati.

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

- ✓ Diagramma della separazione dell'accesso ai dati
- ✓ Decisione sulla durata del DbContext
- ✓ Classi di comando e di risultato
- ✓ Esempio di transazione
- ✓ Tabella di mappatura degli errori

Perché è fatto così

I lab del corso intermedio possono usare servizi fittizi. Le applicazioni avanzate devono collegarsi ad archivi dati reali senza trasformare le schermate in client SQL e senza far trapelare nella UI gli aspetti della persistenza.

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
separazione di EF Core	La linea tra il codice della UI ed Entity Framework Core. Le schermate chiamano servizi e comandi; non tengono mai un DbContext e non scrivono LINQ sulle entità.
durata del DbContext	Quanto a lungo vive un'istanza di DbContext. Contesti brevi, uno per operazione, evitano tracking obsoleto, crescita della memoria e fughe tra sessioni in un'app a lunga esecuzione.
repository o servizio di query	I repository caricano e salvano gli aggregati per i comandi, mentre i servizi di query restituiscono proiezioni di sola lettura modellate per le schermate. Tenerli separati li mantiene semplici entrambi.
confine transazionale	L'unità di lavoro sul database che deve riuscire o fallire tutta insieme. Appartiene a un servizio o a un comando, non a gestori di evento della UI sparsi.
oggetto comando	Una piccola classe che trasporta l'input di un'operazione di business e viene eseguita da un handler, così l'operazione può essere validata, registrata nei log e testata in modo indipendente.
unit of work	Coordina tutte le modifiche di un'operazione e le conferma insieme, offrendo un unico punto in cui gestire transazioni e change tracking.
mappatura degli errori	Tradurre le eccezioni del database, come violazioni di vincolo ed errori di concorrenza, in errori applicativi significativi che la UI può presentare in sicurezza.

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 EnterpriseOps 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 - "Architettura dei dati reale: EF Core, transazioni, repository e comandi" - del corso LearnWisej "Wisej.NET enterprise: dall'architettura al cloud", 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/azure-docker/specs/module-04.md>

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

1. Progetto "EnterpriseOps", 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: Sostituisci la persistenza fittizia di `WorkOrderService` con un'implementazione basata su EF Core o con un'astrazione repository adatta alla produzione. Implementa le operazioni `Create`, `Update`, `Approve`, `Search` e `Audit` con confini transazionali ed errori mappati.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Diagramma della separazione dell'accesso ai dati
 - Decisione sulla durata del `DbContext`
 - Classi di comando e di risultato
 - Esempio di transazione
 - Tabella di mappatura degli errori
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:

- Un salvataggio si può testare senza creare un Form?
- Che cosa c'è dentro la transazione?
- Come vengono segnalati gli errori di concorrenza e di chiave univoca?

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:

- ✓ Un salvataggio si può testare senza creare un Form?
- ✓ Che cosa c'è dentro la transazione?
- ✓ Come vengono segnalati gli errori di concorrenza e di chiave univoca?

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 `Enterprise Wisej.NET 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.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto EnterpriseOps 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/azure-docker/specs/module-04.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 EnterpriseOps 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.