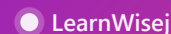


Esercizio di coding con l'IA

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.



L'assistente è la coppia di mani veloci. L'architetto resti tu.

Un assistente può produrre il progetto di questo modulo in pochi minuti. Quello che non può fare è decidere se il risultato è il progetto giusto per la tua applicazione: quella scelta resta tua. Il giudizio lo metti tu: istruiscilo per bene, poi verifica quello che ti torna indietro rispetto a ciò che il Modulo 12 ti ha appena insegnato.

Che cosa stai chiedendo

Crea un pacchetto di deployment per EnterpriseOps con configurazione specifica per ambiente, HealthCheck.json, checklist di smoke test, note sui container di esempio, note sul reverse proxy, indicazioni su sticky session e load balancer e un runbook di rollback.

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

- ✓ Diagramma dell'architettura di deployment
- ✓ Tabella della configurazione degli ambienti
- ✓ HealthCheck.json
- ✓ Runbook di rilascio
- ✓ Checklist di rollback e smoke test

Perché è fatto così

Le applicazioni Wisej.NET avanzate devono andare oltre le macchine degli sviluppatori, verso ambienti ripetibili con disciplina nella configurazione, health check, bilanciamento del carico, rollback e responsabilità operativa.

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
IIS	Il web server di Windows che in molte aziende ospita le applicazioni Wisej.NET, con application pool, regole di riciclo e supporto WebSocket da configurare.
Kestrel	Il web server multiplatforma di ASP.NET Core che esegue direttamente un'applicazione Wisej.NET, di solito dietro un reverse proxy in produzione.
reverse proxy NGINX/Apache	Un server di front-end che termina il TLS, inoltra il traffico WebSocket e instrada le richieste verso una o più istanze dell'applicazione.
Azure/AWS	Piattaforme cloud che ospitano l'applicazione come app service, container o macchine virtuali, ciascuna con il proprio modello di scalabilità e di configurazione.
Docker	Impacchetta l'applicazione e il suo runtime in un'unica immagine, così la stessa build gira in modo identico sulla macchina di uno sviluppatore, su un server di test e nel cloud.
HealthCheck.json	La configurazione dell'health check di Wisej.NET che permette a load balancer e orchestratori di verificare se un'istanza è pronta a servire sessioni.
sticky session	L'affinità del load balancer che instrada ogni richiesta di un utente verso la stessa istanza del server, necessaria perché lo stato di sessione vive nella memoria del server.
CI/CD	Pipeline automatiche di build, test e deployment che trasformano ogni merge in un rilascio ripetibile invece che in una procedura manuale soggetta a errori.

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 12. Puoi aprire tu stesso [module-12.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 12 – "Cloud, container, bilanciamento del carico e ingegneria dei rilasci" – 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-12.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: Crea un pacchetto di deployment per EnterpriseOps con configurazione specifica per ambiente, HealthCheck.json, checklist di smoke test, note sui container di esempio, note sul reverse proxy, indicazioni su sticky session e load balancer e un runbook di rollback.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Diagramma dell'architettura di deployment
 - Tabella della configurazione degli ambienti
 - HealthCheck.json
 - Runbook di rilascio
 - Checklist di rollback e smoke test
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:

- Che cosa richiede le sticky session?
- Dove vengono memorizzati i segreti?
- Come fanno le operations a sapere che un nodo non è in salute?

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:

- ✓ Che cosa richiede le sticky session?
- ✓ Dove vengono memorizzati i segreti?
- ✓ Come fanno le operations a sapere che un nodo non è in salute?

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 12 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 12 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Enterprise Wisej.NET Course/Module 12`.



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 EnterpriseOps 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 EnterpriseOps 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 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."*



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.