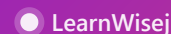


Esercizio di coding con l'IA

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.



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 base della soluzione EnterpriseOps Command Center. Aggiungi progetti o cartelle per UI, controlli condivisi, dominio, servizi applicativi, accesso ai dati, integrazioni, sicurezza, diagnostica, risorse e deployment. Aggiungi una cartella per gli Architecture Decision Record e scrivi l'ADR-001 per la struttura di soluzione scelta.

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

- ✓ Screenshot o albero della struttura di soluzione e cartelle
- ✓ ADR-001 con la decisione sulla struttura della soluzione
- ✓ Pagina degli standard di codifica del team
- ✓ Esempio di denominazione di una schermata di riferimento e di incapsulamento nei servizi
- ✓ Checklist di code review applicata alla prima schermata

Perché è fatto così

Il lavoro avanzato con Wisej.NET non riguarda più la possibilità di costruire una schermata. Riguarda la capacità di un team di far evolvere una grande applicazione per anni senza trasformare la produttività in debito tecnico nascosto.

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
struttura di una soluzione enterprise	Suddivide la soluzione in progetti di UI, servizi applicativi, dominio e infrastruttura, così le schermate restano sottili e le regole di business hanno un'unica sede.
architecture decision record	Documenti brevi che registrano contesto, opzioni, decisione e conseguenze di una scelta progettuale, così il team può rivederla e reconsiderarla in seguito.
suddivisioni compatibili con il designer	Mantiene form e pagine modificabili nel designer, mentre flusso di lavoro, validazione e accesso ai dati passano a servizi che il designer non tocca mai.
gate di revisione	Punti di controllo definiti in cui codice, architettura e sicurezza vengono rivisti prima che una funzionalità venga integrata, così la qualità è imposta e non solo sperata.
convenzioni del team	Regole condivise di denominazione, cartelle e registrazione dei servizi che rendono ogni schermata familiare a qualsiasi sviluppatore del team.
controlli sul debito tecnico	Pratiche come i registri del debito, i budget di refactoring e le regole di deprecazione, che mantengono le scorciatoie visibili e pianificate invece di dimenticarle.

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 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 1 - "Architettura Wisej.NET enterprise, leadership tecnica e governance" - 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-01.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 la base della soluzione EnterpriseOps Command Center. Aggiungi progetti o cartelle per UI, controlli condivisi, dominio, servizi applicativi, accesso ai dati, integrazioni, sicurezza, diagnostica, risorse e deployment. Aggiungi una cartella per gli Architecture Decision Record e scrivi l'ADR-001 per la struttura di soluzione scelta.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Screenshot o albero della struttura di soluzione e cartelle
 - ADR-001 con la decisione sulla struttura della soluzione
 - Pagina degli standard di codifica del team
 - Esempio di denominazione di una schermata di riferimento e di incapsulamento nei servizi
 - Checklist di code review applicata alla prima schermata
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 nuovo sviluppatore senior riesce a trovare il servizio di workflow di una schermata in meno di un minuto?
- Il team sa spiegare perché esiste un confine?
- Il designer è ancora utilizzabile dopo la suddivisione architetturale?

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 nuovo sviluppatore senior riesce a trovare il servizio di workflow di una schermata in meno di un minuto?
- ✓ Il team sa spiegare perché esiste un confine?
- ✓ Il designer è ancora utilizzabile dopo la suddivisione architetturale?

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 `Enterprise Wisej.NET 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.

- ✓ **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."*
- ✓ **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."*



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.