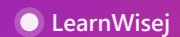


Esercizio di coding con l'IA

Costruisci il progetto OpsMonitor 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.



Lo stesso modulo, costruito come costruirai davvero le cose.

Tutte le altre sezioni di questo modulo spiegano come funziona "Controlli dati, payload del browser e grandi superfici di UI". Questa ti chiede di produrlo: dirigendo un assistente con la specifica del modulo stesso, poi verificando che il risultato giri e faccia quello che descrive la traccia del lab. È il flusso di lavoro che userai sul lavoro vero, provato su qualcosa di cui conosci già la risposta giusta.

Che cosa stai chiedendo

Riduci le due superfici più grandi di WisejPerfLab. La griglia dei ticket collega 50.000 entità complete con le proprietà di navigazione: sostituisci il binding con una proiezione TicketGridRow delle sole colonne visibili, imposta VirtualMode e RowCount e fornisci le celle da un gestore CellValueNeeded appoggiato a un servizio di query paginate. L'albero dei clienti costruisce tutti i nodi all'avvio: carica invece i figli all'espansione, usando un segnaposto con il conteggio per i rami chiusi. Misura ogni schermata prima e dopo con CPU Usage e .NET Object Allocation sul server e con il pannello di rete del browser per la dimensione del payload degli aggiornamenti.

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

- ✓ Proiezione TicketGridRow collegata al posto delle entità complete, con le sole colonne visualizzate
- ✓ Griglia in virtual mode con RowCount e un gestore CellValueNeeded servito da un servizio di query paginate
- ✓ Albero dei clienti che carica i nodi figli all'espansione con un segnaposto con il conteggio per i rami chiusi
- ✓ Numeri di CPU e allocazioni prima/dopo per gli scenari di caricamento della griglia ed espansione dell'albero
- ✓ Prove di rete del browser che confrontano la dimensione del payload degli aggiornamenti prima e dopo, con l'avvertenza sulla compressione

Perché è fatto così

Le grandi superfici di UI sono la fonte più comune di problemi di prestazioni in Wisej.NET, e sono il contenitore che il profiler non riesce a vedere del tutto. La griglia dei ticket di WisejPerfLab collega cinquantamila entità con le loro proprietà di navigazione; l'albero dei clienti costruisce tutti i nodi all'avvio.

Su un desktop quel pattern era accettabile, perché il controllo e i dati vivevano nello stesso processo della schermata. Qui lo stesso codice paga quattro volte: memoria del server per trattenere gli oggetti, CPU del server per costruire e serializzare lo stato della UI, rete per trasportare l'aggiornamento e tempo del browser per renderlo. Questo modulo è quello in cui la migliore ottimizzazione di solito non è un algoritmo più veloce ma meno UI che esiste contemporaneamente.

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
riga di proiezione	Un piccolo tipo che contiene solo le colonne visualizzate da una schermata, collegato al posto del grafo delle entità, così né il server né il browser trasportano dati che nessuno guarda.
virtual mode	La griglia mantiene un numero di righe invece di oggetti riga e chiede i valori delle celle quando servono, il che elimina gli oggetti per riga e il rendering delle righe fino a cui nessuno ha scorso.
CellValueNeeded	L'evento che fornisce un valore su richiesta in virtual mode; gira molto spesso, quindi è un percorso critico per costruzione e deve restituire rapidamente.
paginazione e filtri lato server	Chiedere al database la pagina che l'utente sta guardando, invece di caricare tutto e filtrare nel codice di UI.
caricamento lazy dei nodi	Costruire i figli dell'albero quando un ramo viene espanso, con un segnaposto con il conteggio finché è chiuso, invece di materializzare l'intera gerarchia all'avvio.
payload degli aggiornamenti	La differenza serializzata che il server invia dopo un gestore; payload grandi o frequenti costano tempo di rete e del browser che nessuno strumento di Visual Studio riporta.
la compressione non è lavoro gratis	Wisej.NET comprime il traffico HTTP e WebSocket, il che riduce i byte ma non rende economico produrre, trasmettere o rendere un aggiornamento inutile.

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 5. Puoi aprire tu stesso [module-05.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



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece

Codex; qualsiasi piano che possa navigare il web recupererà la specifica.

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 OpsMonitor 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 5 - "Controlli dati, payload del browser e grandi superfici di UI" - del corso LearnWisej "Prestazioni e profiling", 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/performance-and-profiling/specs/module-05.md>

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

1. Progetto "OpsMonitor", 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: Riduci le due superfici più grandi di WisejPerfLab. La griglia dei ticket collega 50.000 entità complete con le proprietà di navigazione: sostituisci il binding con una proiezione `TicketGridRow` delle sole colonne visibili, imposta `VirtualMode` e `RowCount` e fornisci le celle da un gestore `CellValueNeeded` appoggiato a un servizio di query paginate. L'albero dei clienti costruisce tutti i nodi all'avvio: carica invece i figli all'espansione, usando un segnaposto con il conteggio per i rami chiusi. Misura ogni schermata prima e dopo con CPU Usage e .NET Object Allocation sul server e con il pannello di rete del browser per la dimensione del payload degli aggiornamenti.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Proiezione `TicketGridRow` collegata al posto delle entità complete, con le sole colonne visualizzate
 - Griglia in virtual mode con `RowCount` e un gestore `CellValueNeeded` servito da un servizio di query paginate
 - Albero dei clienti che carica i nodi figli all'espansione con un segnaposto con il conteggio per i rami chiusi
 - Numeri di CPU e allocazioni prima/dopo per gli scenari di caricamento della griglia ed espansione dell'albero
 - Prove di rete del browser che confrontano la dimensione del payload degli aggiornamenti prima e dopo, con l'avvertenza sulla compressione
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:

- Il virtual mode ha eliminato gli oggetti riga ma la tua traccia di CPU è migliorata appena. Che cosa ti dice questo su dove si trovava davvero il costo, e quale misura guarderesti dopo?
- Il tuo gestore `CellValueNeeded` è corretto ma lento durante uno scorrimento rapido. Elenca tre cose che potrebbe fare e che un gestore per cella non deve mai fare.
- Il payload compresso dell'espansione dell'albero è cambiato appena dopo il caricamento

lazy, ma gli utenti dicono che la schermata sembra più veloce. Fornisci una spiegazione plausibile che il solo conteggio dei byte nasconderebbe.

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:

- ✓ Il virtual mode ha eliminato gli oggetti riga ma la tua traccia di CPU è migliorata appena. Che cosa ti dice questo su dove si trovava davvero il costo, e quale misura guarderesti dopo?
- ✓ Il tuo gestore CellValueNeeded è corretto ma lento durante uno scorrimento rapido. Elenca tre cose che potrebbe fare e che un gestore per cella non deve mai fare.
- ✓ Il payload compresso dell'espansione dell'albero è cambiato appena dopo il caricamento lazy, ma gli utenti dicono che la schermata sembra più veloce. Fornisci una spiegazione plausibile che il solo conteggio dei byte nasconderebbe.

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 5 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 5 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Performance and Profiling Course/Module 5`.



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.

- ✓ **Fallo sopravvivere a una brutta giornata.** *"Esamina questo progetto OpsMonitor 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 OpsMonitor 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/performance-and-profiling/specs/module-05.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."*



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.