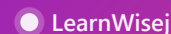


Esercizio di coding con l'IA

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.



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

Esegui il profiling dei due scenari dominati dalle attese in WisejPerfLab. Esegui la ricerca dei ticket sotto lo strumento Database e abbina ogni query all'azione di UI: il caricamento della griglia lancia una query sul cliente per ogni riga visualizzata e la query dei dettagli preleva grandi colonne di testo che nessuno mostra. Sostituisci entrambe con un'unica query di proiezione no-tracking che seleziona solo le colonne della griglia e pagina il risultato. Poi esegui l'esportazione CSV sotto File I/O e .NET Async, trova la chiamata bloccante .Result nel gestore del clic e le piccole scritture ripetute, e sposta la generazione in un flusso in background con Application.StartTask che scrive il file in streaming e segnala l'avanzamento in modo limitato invece di inviare ogni singolo passo.

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

- ✓ Traccia Database che abbina le query alle azioni di UI, con il numero di query N+1 prima della correzione
- ✓ Un'unica query di proiezione no-tracking con paginazione al posto delle query per riga e di quelle che prelevano troppo
- ✓ Prove File I/O e .NET Async per l'esportazione, che indicano la chiamata bloccante e il pattern di scrittura
- ✓ Esportazione spostata in un task in background che scrive in streaming e invia l'avanzamento solo nei passi significativi
- ✓ Numero di query, durata delle query e tempo reale dell'esportazione prima/dopo, con i rischi residui

Perché è fatto così

Il Modulo 2 ha misurato 1.840 ms di tempo trascorso e ha trovato solo 300 ms di CPU. Il Modulo 3 ha eliminato la maggior parte di quella CPU e la schermata non è ancora veloce, perché i 600 ms mancanti

non sono mai stati calcolati — erano attesi. È la categoria meno diagnosticata in un'applicazione gestionale: il server è inattivo, il grafico della CPU è piatto e l'utente sta guardando un indicatore di caricamento. Da qualche parte una query gira una volta per riga, un form di dettaglio preleva una colonna che nessuno visualizza, oppure un gestore del clic è bloccato su `Result` mentre un thread del thread pool non fa nulla. Gli strumenti che vedono tutto questo sono Database, File I/O, .NET Async e Instrumentation, e questo modulo è quello in cui la latenza residua di WisejPerfLab riceve finalmente un nome.

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
strumento Database	Registra le query ADO.NET ed EF Core lanciate da uno scenario, con le loro durate e il loro ordine, così ogni query può essere abbinata all'azione di UI che l'ha causata.
query N+1	Una query per elencare, poi una in più per ogni elemento restituito; l'elenco sembra a posto con dieci righe e crolla con mille, ed è invisibile al profiling della CPU.
prelievo eccessivo	Selezionare colonne o righe che la schermata non mostra mai — le grandi colonne di testo e binarie sono le colpevoli abituali, e costano tempo di query, trasferimento e memoria.
query no-tracking	Una query in sola lettura che non attiva il change tracking, adatta a griglie ed elenchi che visualizzano dati invece di modificarli.
strumento.NET Async	Mostra le attività async e dove restano in attesa, ed è così che una catena involontariamente sequenziale o bloccata diventa visibile.
bloccare su async	Chiamare <code>Result</code> o <code>Wait()</code> da un gestore; trattiene un thread durante l'attesa e, sotto carico, produce sintomi che sembrano esaurimento del thread pool.
esportazione in background	La generazione lunga di file o report spostata fuori dal percorso immediato del clic con <code>Application.StartTask</code> , segnalando l'avanzamento a intervalli significativi invece che a ogni passo.

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 6. Puoi aprire tu stesso [module-06.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 6 - "Database, I/O su file, async e attese esterne" - 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-06.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: Esegui il profiling dei due scenari dominati dalle attese in `WisejPerfLab`. Esegui la ricerca dei ticket sotto lo strumento Database e abbinare ogni query all'azione di UI: il caricamento della griglia lancia una query sul cliente per ogni riga visualizzata e la query dei dettagli preleva grandi colonne di testo che nessuno mostra. Sostituisci entrambe con un'unica query di proiezione no-tracking che seleziona solo le colonne della griglia e pagina il risultato. Poi esegui l'esportazione CSV sotto File I/O e .NET Async, trova la chiamata bloccante `.Result` nel gestore del clic e le piccole scritture ripetute, e sposta la generazione in un flusso in background con `Application.StartTask` che scrive il file in streaming e segnala l'avanzamento in modo limitato invece di inviare ogni singolo passo.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Traccia Database che abbinare le query alle azioni di UI, con il numero di query N+1 prima della correzione
 - Un'unica query di proiezione no-tracking con paginazione al posto delle query per riga e di quelle che prelevano troppo
 - Prove File I/O e .NET Async per l'esportazione, che indicano la chiamata bloccante e il pattern di scrittura
 - Esportazione spostata in un task in background che scrive in streaming e invia l'avanzamento solo nei passi significativi
 - Numero di query, durata delle query e tempo reale dell'esportazione prima/dopo, con i rischi residui
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:

- La ricerca ora lancia una query invece di duecentouno, ma il tempo trascorso si è solo dimezzato. Dove si trova il resto del tempo, e quale strumento useresti per scoprirlo?
- Un collega dice che l'esportazione va bene perché `.Result` "blocca solo un thread". Spiega che cosa succede con cinquanta esportazioni concorrenti e quali schermate l'utente noterebbe rallentare.

– Tra le cose che questo lab mette in cache, quale è sicura da condividere tra sessioni e quale no, e quale regola hai usato per decidere?

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:

- ✓ La ricerca ora lancia una query invece di duecentouno, ma il tempo trascorso si è solo dimezzato. Dove si trova il resto del tempo, e quale strumento useresti per scoprirlo?
- ✓ Un collega dice che l'esportazione va bene perché.Result "blocca solo un thread". Spiega che cosa succede con cinquanta esportazioni concorrenti e quali schermate l'utente noterebbe rallentare.
- ✓ Tra le cose che questo lab mette in cache, quale è sicura da condividere tra sessioni e quale no, e quale regola hai usato per decidere?

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 6 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 6 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Performance and Profiling Course/Module 6`.



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 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-06.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 OpsMonitor 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.