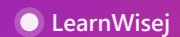


Esercizio di coding con l'IA

Costruisci il progetto TicketOpsLive 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 prima schermata di TicketOps Live: una pagina con una barra di stato dal vivo, una label con l'orologio, un indicatore di avanzamento e un heartbeat simulato del server che aggiorna il browser senza alcun clic dell'utente. Criteri di accettazione: la UI si aggiorna una volta al secondo dopo che lo studente fa clic su Start; il browser mostra le modifiche senza clic aggiuntivi; Stop disattiva il ciclo entro un secondo; avviare due volte non crea due cicli in concorrenza; il codice verifica se la pagina è stata eliminata (disposed).

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

- ✓ Aggiungi un pannello in alto chiamato statusPanel.
- ✓ Aggiungi le label clockLabel, connectionLabel e activityLabel.
- ✓ Aggiungi una progress bar chiamata serverLoadBar.
- ✓ Aggiungi i pulsanti startButton e stopButton.
- ✓ Implementa un'attività in background che simula l'heartbeat quando si fa clic su startButton.
- ✓ L'heartbeat deve aggiornare l'orologio, un valore di carico fittizio e un messaggio di attività una volta al secondo.
- ✓ Ferma gli aggiornamenti quando si fa clic su stopButton o quando la pagina viene eliminata (disposed).

Perché è fatto così

Le applicazioni web in tempo reale si associano di solito a chat, giochi, quotazioni di borsa, schermate di monitoraggio e dashboard. Nei sistemi gestionali la stessa esigenza compare in forme meno spettacolari ma più preziose: un supervisore vede i nuovi ticket appena arrivano, un operatore di produzione vede una macchina fermarsi, un dispatcher vede che un intervento è stato riassegnato, oppure un addetto alla fatturazione vede che un'importazione è ancora in corso senza premere Aggiorna.

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 TicketOpsLive 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 web in tempo reale in Wisej.NET" - del corso LearnWisej "App in tempo reale con server push", 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/realtime-server-push/specs/module-01.md>

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

1. Progetto "TicketOpsLive", 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 prima schermata di TicketOps Live: una pagina con una barra di stato dal vivo, una label con l'orologio, un indicatore di avanzamento e un heartbeat simulato del server che aggiorna il browser senza alcun clic dell'utente. Criteri di accettazione: la UI si aggiorna una volta al secondo dopo che lo studente fa clic su Start; il browser mostra le modifiche senza clic aggiuntivi; Stop disattiva il ciclo entro un secondo; avviare due volte non crea due cicli in concorrenza; il codice verifica se la pagina è stata eliminata (disposed).
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Aggiungi un pannello in alto chiamato `statusPanel`
 - Aggiungi le label `clockLabel`, `connectionLabel` e `activityLabel`
 - Aggiungi una progress bar chiamata `serverLoadBar`
 - Aggiungi i pulsanti `startButton` e `stopButton`
 - Implementa un'attività in background che simula l'heartbeat quando si fa clic su `startButton`
 - L'heartbeat deve aggiornare l'orologio, un valore di carico fittizio e un messaggio di attività una volta al secondo
 - Ferma gli aggiornamenti quando si fa clic su `stopButton` o quando la pagina viene eliminata (disposed)
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:

- Perché questo lab chiama `Application.Update(this)` dentro il ciclo?
- Che cosa cambia se `WebSocket` non è disponibile?
- Perché `_running` non è un campo statico?
- Quale sarebbe il rischio di aggiornare ogni 10 millisecondi?

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:

- ✓ Perché questo lab chiama `Application.Update(this)` dentro il ciclo?
- ✓ Che cosa cambia se `WebSocket` non è disponibile?
- ✓ Perché `_running` non è un campo statico?
- ✓ Quale sarebbe il rischio di aggiornare ogni 10 millisecondi?

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 `Real-Time Server Push 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.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto TicketOpsLive 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/realtime-server-push/specs/module-01.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 TicketOpsLive 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.