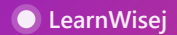


Esercizio di coding con l'IA

Costruisci il progetto FieldOps 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.



Istruiscilo, leggilo, poi mettilo alla prova.

Questo esercizio ha tre parti, e solo la prima riguarda il prompt. Istruisci un assistente con la specifica del modulo e ti fai restituire una soluzione FieldOps; leggi quello che ha scritto e decidi se fa davvero ciò che il modulo chiedeva; poi lo spingi più in là di quanto il modulo chiedesse. La seconda e la terza parte sono dove si impara.

Che cosa stai chiedendo

Crea il pacchetto di FieldOps per Windows: uno shell WebView2 con finestra nativa, titolo personalizzato e icona nell'area di notifica, un bridge che permette all'app di aprire un selettore di cartelle nativo per esportare i report degli ordini di lavoro e di mostrare una notifica nativa quando arriva un nuovo ordine, una configurazione che punta lo shell al server remoto con un ripiego locale, e un installer con un controllo degli aggiornamenti all'avvio.

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

- ✓ Shell desktop WebView2 con finestra e icona nell'area di notifica
- ✓ Bridge nativo per selettore di cartelle e notifiche
- ✓ Configurazione del server remoto e locale
- ✓ Pacchetto di installazione
- ✓ Controllo degli aggiornamenti all'avvio

Perché è fatto così

I tecnici di FieldOps passano un intero turno dentro l'app: un portatile nel furgone, una docking station al deposito. La build per il browser dei primi due moduli fa il suo lavoro, ma non ha un'icona nel menu Start, non ha presenza nell'area di notifica quando viene assegnato un intervento, non ha modo di salvare un report esportato in una cartella scelta dal tecnico, e non offre nulla che l'IT possa distribuire attraverso il proprio canale software abituale. Questo modulo avvolge quella stessa applicazione, senza modificarla, in uno shell Windows che fornisce esattamente queste quattro cose.

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
controllo WebView2	Il motore Chromium che Microsoft distribuisce come Evergreen Runtime ridistribuibile, incorporato in una finestra WinForms o WPF così che la tua pagina venga resa esattamente come in Edge.
shell desktop	Un piccolo eseguibile nativo che possiede la finestra, l'icona nell'area di notifica e le chiamate al sistema operativo, e punta un controllo WebView2 alla tua applicazione Wisej.NET.
cornice della finestra	Il titolo, l'icona, la dimensione minima, la posizione ricordata e il comportamento a istanza singola che fanno sembrare l'app installata invece che incorporata.
bridge dei messaggi	Il canale bidirezionale tra pagina e shell: PostWebMessageAsJson verso la pagina, WebMessageReceived verso lo shell, oppure un host object pubblicato con AddHostObjectToScript.
interop JavaScript	Il modo in cui il C# lato server raggiunge quel canale: Application.Eval e Application.Call eseguono script nel contesto browser della sessione e la risposta torna come evento client.
topologia del server	Se lo shell naviga verso un server aziendale remoto o avvia un processo Wisej.NET self-hosted su loopback — deciso all'avvio e registrato nel log, mai indovinato.
canale di aggiornamento	Un endpoint della versione che lo shell interroga all'avvio, più l'installer che scarica quando la risposta è più recente della versione in esecuzione.

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 3. Puoi aprire tu stesso [module-03.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 FieldOps 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 3 - "Pacchetti desktop con uno shell WebView" - del corso LearnWisej "Pacchetti ibridi e nativi", 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/hybrid-native-packaging/specs/module-03.md>

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

1. Progetto "FieldOps", 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 il pacchetto di FieldOps per Windows: uno shell WebView2 con finestra nativa, titolo personalizzato e icona nell'area di notifica, un bridge che permette all'app di aprire un selettore di cartelle nativo per esportare i report degli ordini di lavoro e di mostrare una notifica nativa quando arriva un nuovo ordine, una configurazione che punta lo shell al server remoto con un ripiego locale, e un installer con un controllo degli aggiornamenti all'avvio.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Shell desktop WebView2 con finestra e icona nell'area di notifica
 - Bridge nativo per selettore di cartelle e notifiche
 - Configurazione del server remoto e locale
 - Pacchetto di installazione
 - Controllo degli aggiornamenti all'avvio
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 tecnico avvia due esportazioni a un secondo di distanza e lo shell risponde a entrambe le finestre di dialogo delle cartelle. Che cosa nel progetto del tuo bridge impedisce che il secondo percorso venga scritto nel primo report, e che cosa andrebbe storto senza?
- Il tuo pulsante Export ora apre una finestra di dialogo nativa per le cartelle. Descrivi con precisione che cosa fa lo stesso pulsante quando la pagina viene aperta in Edge invece che nello shell, e in quale punto del tuo codice viene decisa quella differenza.
- L'endpoint della versione non è raggiungibile perché il furgone non ha segnale. Argomenta che cosa dovrebbe fare lo shell a quell'avvio, e spiega che cosa perde un tecnico con ciascuna delle alternative che hai scartato.

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 tecnico avvia due esportazioni a un secondo di distanza e lo shell risponde a entrambe le finestre di dialogo delle cartelle. Che cosa nel progetto del tuo bridge impedisce che il secondo percorso venga scritto nel primo report, e che cosa andrebbe storto senza?
- ✓ Il tuo pulsante Export ora apre una finestra di dialogo nativa per le cartelle. Descrivi con precisione che cosa fa lo stesso pulsante quando la pagina viene aperta in Edge invece che nello shell, e in quale punto del tuo codice viene decisa quella differenza.
- ✓ L'endpoint della versione non è raggiungibile perché il furgone non ha segnale. Argomenta che cosa dovrebbe fare lo shell a quell'avvio, e spiega che cosa perde un tecnico con ciascuna delle alternative che hai scartato.

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.



Per questo corso non c'è ancora un esempio di riferimento — il tuo è il primo

Il [repository degli esempi LearnWisej](#) contiene build già fatte per quasi tutti i corsi, ma non ancora per questo. Significa che non c'è niente con cui confrontarsi e nessuna scorciatoia: la specifica e la tua revisione sono tutta la rete di sicurezza. Tieni quello che costruisci — e se viene bene, ci farebbe piacere vederlo.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito e prova questi. Ognuno è un esercizio di revisione ed estensione, non una riscrittura.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto FieldOps 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/hybrid-native-packaging/specs/module-03.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 FieldOps 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.