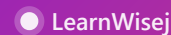


Esercizio di coding con l'IA

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.



Scrivere il codice è ormai la metà facile.

Ottenere codice funzionante per "Shell mobile con .NET MAUI" da un assistente moderno richiede un buon prompt. Giudicarlo, correggerlo ed estenderlo richiede tutto quello che questo modulo ha trattato. È su questa divisione che è costruito l'esercizio: tu metti la specifica e la revisione, l'assistente mette la digitazione.

Che cosa stai chiedendo

Costruisci lo shell mobile di FieldOps con .NET MAUI: una pagina ibrida che ospita l'app su Android e iOS, una sessione che sopravvive al passaggio in background e riprende al ritorno, il pulsante Indietro di Android mappato sulla navigazione interna dell'app, un padding di safe area per i dispositivi con notch, e una build di debug in esecuzione su un emulatore per piattaforma con gli screenshot dell'elenco degli ordini di lavoro.

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

- ✓ Progetto di shell ibrido.NET MAUI per Android e iOS
- ✓ Gestione del ciclo di vita per background e ripresa
- ✓ Mappatura del pulsante Indietro e della safe area
- ✓ Build sugli emulatori per entrambe le piattaforme
- ✓ Screenshot da entrambi gli emulatori

Perché è fatto così

I tecnici di FieldOps lavorano da un telefono in una sala macchine, in uno scantinato o in un furgone, e una scheda del browser è un contenitore scadente per questo: nessuna icona nella schermata home da distribuire, nessuno store o canale enterprise, e nessuna strada verso la fotocamera e il GPS di cui avrà bisogno il Modulo 5. Il sistema operativo mobile è anche un ospite molto più aggressivo — sospende l'app nell'istante in cui arriva una chiamata, la ridisegna sotto un notch e consegna al tecnico un gesto Indietro di sistema che FieldOps non ha mai chiesto.

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
shell ibrido.NET MAUI	Una piccola app nativa per Android e iOS la cui unica pagina ospita la tua applicazione Wisej.NET in una web view, così che il tecnico veda proprio le schermate C# che già rilasci.
host web view nativo	Il motore browser della piattaforma incorporato in quella pagina — Android System WebView, WKWebView su iOS — che esegue lo stesso client che riceve un browser desktop, ed è per questo che nessuna schermata va riscritta per il mobile.
bridge ibrido	Il canale bidirezionale tra la pagina ospitata e il codice nativo, così che il C# lato server possa chiedere qualcosa al dispositivo e ricevere la risposta come evento invece che come navigazione di pagina.
ciclo di vita dell'applicazione MAUI	Gli eventi che il sistema operativo genera attorno all'app — finestra creata, attivata, disattivata, fermata e ripresa, più OnSleep e OnResume — nessuno dei quali viene generato da una scheda del browser.
finestra di sopravvivenza della sessione	L'intervallo di tempo in background durante il quale la sessione Wisej.NET è ancora viva sul server, così che una ripresa possa riconnettersi allo stato esistente invece di far ripartire l'app da zero.
inset della safe area	Il padding specifico di ogni dispositivo che tiene i contenuti lontani dal notch, dalla barra di stato e dalla striscia dell'indicatore home, comunicato dalla piattaforma invece che dedotto dalla dimensione dello schermo.
mappatura della navigazione Indietro	La regola che decide che cosa fa il gesto Indietro di Android: tornare indietro nella navigazione interna dell'app, oppure chiedere conferma prima di uscire — mai distruggere in silenzio un ordine di lavoro compilato a metà.

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 4. Puoi aprire tu stesso [module-04.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.

Il prompt, per intero — leggilo, poi modificalo se vuoi un progetto diverso

Sto seguendo il Modulo 4 - "Shell mobile con .NET MAUI" - 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-04.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: Costruisci lo shell mobile di FieldOps con .NET MAUI: una pagina ibrida che ospita l'app su Android e iOS, una sessione che sopravvive al passaggio in background e riprende al ritorno, il pulsante Indietro di Android mappato sulla navigazione interna dell'app, un padding di safe area per i dispositivi con notch, e una build di debug in esecuzione su un emulatore per piattaforma con gli screenshot dell'elenco degli ordini di lavoro.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Progetto di shell ibrido.NET MAUI per Android e iOS
 - Gestione del ciclo di vita per background e ripresa
 - Mappatura del pulsante Indietro e della safe area
 - Build sugli emulatori per entrambe le piattaforme
 - Screenshot da entrambi gli emulatori
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 è a metà di un aggiornamento dello stato quando arriva una chiamata di tre minuti. Che cosa deve essere vero sul server e nello shell perché l'aggiornamento lasciato a metà sia ancora lì dopo, e che cosa cambia se la chiamata dura un'ora?
- Lo shell non sa nulla di quale schermata di FieldOps sia aperta. Come fa un gesto Indietro a scegliere tra chiudere la pagina di dettaglio, chiudere una finestra di dialogo e uscire dall'app, e dove vive davvero quello stato?
- Quali modifiche di questo modulo noterebbe ancora un tecnico se aprisse FieldOps nel browser del telefono invece che nello shell, e quali sparirebbero? Che cosa ti dice questo su dove mettere il padding della safe area?

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 è a metà di un aggiornamento dello stato quando arriva una chiamata di tre minuti. Che cosa deve essere vero sul server e nello shell perché l'aggiornamento lasciato a metà sia ancora lì dopo, e che cosa cambia se la chiamata dura un'ora?
- ✓ Lo shell non sa nulla di quale schermata di FieldOps sia aperta. Come fa un gesto Indietro a scegliere tra chiudere la pagina di dettaglio, chiudere una finestra di dialogo e uscire dall'app, e dove vive davvero quello stato?
- ✓ Quali modifiche di questo modulo noterebbe ancora un tecnico se aprisse FieldOps nel browser del telefono invece che nello shell, e quali sparirebbero? Che cosa ti dice questo su dove mettere il padding della safe area?

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.

- ✓ **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-04.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."*

- ✓ **Sostituisci un pezzo senza spostare le giunzioni.** *"Scegli la parte di questo progetto FieldOps 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."*



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.