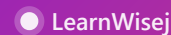


Esercizio di coding con l'IA

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.



L'assistente è la coppia di mani veloci. L'architetto resti tu.

Un assistente può produrre il progetto di questo modulo in pochi minuti. Quello che non può fare è decidere se il risultato è il progetto giusto per la tua applicazione: quella scelta resta tua. Il giudizio lo metti tu: istruiscilo per bene, poi verifica quello che ti torna indietro rispetto a ciò che il Modulo 2 ti ha appena insegnato.

Che cosa stai chiedendo

Trasforma FieldOps in una PWA: aggiungi il manifest con icone, colore del tema e visualizzazione standalone, registra un service worker che mette in cache lo shell e le risorse statiche, mostra un invito all'installazione dopo la prima sessione riuscita, visualizza una pagina offline con il riepilogo degli ultimi ordini di lavoro sincronizzati quando il server non è raggiungibile, e implementa il flusso di aggiornamento che ricarica l'app quando è disponibile una nuova versione.

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

- ✓ Manifest dell'app web con icone e colori
- ✓ Service worker che mette in cache lo shell
- ✓ Invito all'installazione dopo la prima sessione
- ✓ Pagina offline con l'ultimo riepilogo sincronizzato
- ✓ Flusso di aggiornamento e controllo della versione

Perché è fatto così

I tecnici di FieldOps non aprono un browser per digitare un URL. Toccano un'icona su un telefono rimasto in un furgone tutta la mattina, in una sala macchine dove il segnale sparisce a metà di un intervento. Una progressive web app dà loro quell'icona, una finestra senza la cornice del browser e una pagina che dice ancora qualcosa di utile quando la rete scompare — ma FieldOps è un'app Wisej.NET, quindi la sua interfaccia utente vive sul server, e fingere il contrario è il modo in cui una PWA si trasforma in un ticket di assistenza.

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
manifest dell'app web	Un piccolo file JSON, collegato dalla pagina di avvio, che comunica al browser nome, icone, colori, modalità di visualizzazione e start URL dell'app — senza di esso il browser non ha nulla da installare.
modalità di visualizzazione	Il membro display del manifest. standalone avvia FieldOps in una finestra propria senza barra degli indirizzi, ed è ciò che fa smettere a un'app installata di sembrare un segnalibro.
icone maskable	Icone dichiarate con purpose: "maskable" e un margine di sicurezza, così che Android possa ritagliarle nella propria forma senza tagliare il disegno.
service worker	Uno script che il browser esegue accanto alla pagina, servito dall'origine dell'app stessa su HTTPS, che può rispondere alle richieste della pagina da una cache quando la connessione è lenta o assente.
shell applicativo	I file statici di cui una sessione ha bisogno prima che il server possa dire qualsiasi cosa: la pagina di avvio, gli script client, il tema e le icone. È ciò che il worker ha il permesso di mettere in cache.
cache-first vs network-only	La regola di instradamento nel gestore fetch del worker: i file statici dello shell arrivano prima dalla cache, mentre tutto ciò che la sessione Wisej.NET scambia con il server deve andare direttamente in rete, mai a una copia in cache.
ciclo di aggiornamento	Un nuovo worker si installa mentre quello vecchio controlla ancora la pagina, e poi resta in attesa. skipWaiting, clients.claim e un ricaricamento controllato sono ciò che trasforma quel worker in attesa nella versione che il tecnico esegue davvero.

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 2. Puoi aprire tu stesso [module-02.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 2 - "Configurare una Progressive Web App" - 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-02.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: Trasforma FieldOps in una PWA: aggiungi il manifest con icone, colore del tema e visualizzazione standalone, registra un service worker che mette in cache lo shell e le risorse statiche, mostra un invito all'installazione dopo la prima sessione riuscita, visualizza una pagina offline con il riepilogo degli ultimi ordini di lavoro sincronizzati quando il server non è raggiungibile, e implementa il flusso di aggiornamento che ricarica l'app quando è disponibile una nuova versione.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Manifest dell'app web con icone e colori
 - Service worker che mette in cache lo shell
 - Invito all'installazione dopo la prima sessione
 - Pagina offline con l'ultimo riepilogo sincronizzato
 - Flusso di aggiornamento e controllo della versione
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 revisore chiede perché il service worker non mette semplicemente in cache ogni risposta, così che FieldOps continui a funzionare sottoterra. Spiega che cosa farebbe davvero alla schermata del tecnico una risposta di sessione Wisej.NET riprodotta, e dove si trova il confine onesto.
- La tua offerta di installazione funziona alla tua scrivania ma non compare mai sul telefono del tecnico. Indica tre condizioni distinte che potrebbero esserne responsabili, e di' come le distingueresti dal dispositivo.
- Un tecnico ha l'app installata e aperta mentre fai il deploy di una nuova build. Descrivi ogni passaggio tra il tuo deploy e quel telefono che esegue il nuovo shell, e individua quale passaggio fallirebbe in silenzio se il nome della cache non cambiasse mai.

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 revisore chiede perché il service worker non mette semplicemente in cache ogni risposta, così che FieldOps continui a funzionare sottoterra. Spiega che cosa farebbe davvero alla schermata del tecnico una risposta di sessione Wisej.NET riprodotta, e dove si trova il confine onesto.
- ✓ La tua offerta di installazione funziona alla tua scrivania ma non compare mai sul telefono del tecnico. Indica tre condizioni distinte che potrebbero esserne responsabili, e di' come le distingueresti dal dispositivo.
- ✓ Un tecnico ha l'app installata e aperta mentre fai il deploy di una nuova build. Descrivi ogni passaggio tra il tuo deploy e quel telefono che esegue il nuovo shell, e individua quale passaggio fallirebbe in silenzio se il nome della cache non cambiasse mai.

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.

- ✓ **Fallo sopravvivere a una brutta giornata.** *"Esamina questo progetto FieldOps 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 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-02.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.