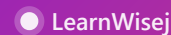


Esercizio di coding con l'IA

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.



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 soluzione FieldOps con un elenco degli ordini di lavoro, una pagina di dettaglio dell'ordine e un flusso di aggiornamento dello stato che funzionano in un semplice browser. Scrivi il registro delle decisioni di distribuzione: confronta browser, PWA, shell desktop e shell mobile rispetto alle esigenze dei tecnici, scegli i target e il loro ordine, e definisci le verifiche delle capacità che l'app userà per abilitare le funzionalità del dispositivo.

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

- ✓ Soluzione FieldOps con elenco, dettaglio e aggiornamento dello stato
- ✓ Tabella di confronto dei modelli di distribuzione
- ✓ Registro delle decisioni di distribuzione con l'ordine dei target
- ✓ Progetto della verifica delle capacità
- ✓ Diagramma dell'architettura con un server e più shell

Perché è fatto così

FieldOps è un'applicazione per l'assistenza sul campo: i tecnici ricevono ordini di lavoro, acquisiscono foto e firme, raggiungono i siti e lavorano con una connettività a singhiozzo. Costruita come una normale applicazione Wisej.NET gira già in qualsiasi browser, il che basta ai dispatcher in ufficio ma non basta a un tecnico in una sala macchine con una sola tacca di segnale e il telefono in mano con i guanti. La domanda a cui risponde questo modulo non è "possiamo costruirla" ma "come arriva alle persone che la usano" — e quella risposta è una decisione di architettura, non un ripensamento di packaging.

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
modello di distribuzione	La strada che l'applicazione percorre per arrivare all'utente: una scheda del browser, una PWA installata, una finestra desktop o un'app mobile distribuita tramite store. Cambia distribuzione, aggiornamenti e accesso al dispositivo — mai il C# che costruisce la schermata.
motore browser ospitante	Ogni modello rende la stessa UI guidata dal server in un motore browser: il browser dell'utente oppure uno incorporato — WebView2 su Windows, WebKit su macOS e iOS, la WebView di sistema su Android.
Wisej.NET Hybrid	Lo shell del framework basato su .NET MAUI che ospita la tua applicazione Wisej.NET esistente su Android, iOS e desktop e aggiunge un bridge verso funzionalità del dispositivo come fotocamera, geolocalizzazione, file e notifiche.
bridge del dispositivo	Il canale bidirezionale tra il tuo C# lato server e lo shell nativo: il server chiede un'azione sul dispositivo, lo shell la esegue e il risultato torna al tuo codice come evento, non come valore di ritorno.
verifica delle capacità	Un test a runtime di ciò che lo shell corrente sa fare davvero, usato per mostrare, nascondere o sostituire una funzionalità così che una stessa schermata si comporti correttamente in ogni modello di distribuzione.
cache dello shell applicativo	L'HTML statico, il CSS, gli script e le icone che il service worker di una PWA può tenere in locale. È ciò che si carica ancora senza rete — la parte interattiva dell'app, resa dal server, non ne fa parte.
asimmetria degli aggiornamenti	Il server può essere ridistribuito in qualsiasi momento e ogni sessione riceve la nuova build al ricaricamento, mentre un installer desktop o un pacchetto per lo store richiede prima un rilascio e un'installazione.

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 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 1 - "Modelli di hosting oltre il browser" - 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-01.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 la soluzione FieldOps con un elenco degli ordini di lavoro, una pagina di dettaglio dell'ordine e un flusso di aggiornamento dello stato che funzionano in un semplice browser. Scrivi il registro delle decisioni di distribuzione: confronta browser, PWA, shell desktop e shell mobile rispetto alle esigenze dei tecnici, scegli i target e il loro ordine, e definisci le verifiche delle capacità che l'app userà per abilitare le funzionalità del dispositivo.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Soluzione FieldOps con elenco, dettaglio e aggiornamento dello stato
 - Tabella di confronto dei modelli di distribuzione
 - Registro delle decisioni di distribuzione con l'ordine dei target
 - Progetto della verifica delle capacità
 - Diagramma dell'architettura con un server e più shell
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 perde il segnale a metà dell'aggiornamento di un ordine di lavoro. Descrivi esattamente quali parti di FieldOps si possono ancora mostrare da uno shell in cache, quali no, e che cosa deve fare la tua schermata nel momento in cui la connessione cade.
- Il tuo registro delle decisioni mette lo shell Android prima della PWA. Che cosa dovrebbe essere vero per i dispositivi e la rete dei tecnici perché quell'ordine sia sbagliato?
- La pagina di dettaglio ha bisogno di una foto. Spiega perché una verifica delle capacità con un ripiego su Upload è un progetto migliore di un interruttore a compile time che produce una versione della schermata solo per mobile.

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 perde il segnale a metà dell'aggiornamento di un ordine di lavoro. Descrivi esattamente quali parti di FieldOps si possono ancora mostrare da uno shell in cache, quali no, e che cosa deve fare la tua schermata nel momento in cui la connessione cade.
- ✓ Il tuo registro delle decisioni mette lo shell Android prima della PWA. Che cosa dovrebbe essere vero per i dispositivi e la rete dei tecnici perché quell'ordine sia sbagliato?
- ✓ La pagina di dettaglio ha bisogno di una foto. Spiega perché una verifica delle capacità con un ripiego su Upload è un progetto migliore di un interruttore a compile time che produce una versione della schermata solo per mobile.

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.

- ✓ **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."*
- ✓ **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."*



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.