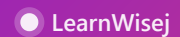


Esercizio di coding con l'IA

Costruisci il progetto AdaptiveOps 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.



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

Crea il tema AdaptiveOps nel Theme Builder partendo da un tema di base integrato. Definisci una palette di colori con nome con brandPrimary, brandAccent, surface, surfaceAlt, textMain, textMuted, danger, warning, success e focusFrame più i font default, heading e mono, poi ridisegna le appearance di button, panel, tab, intestazione della griglia, editor non valido e tooltip usando quei token. Aggiungi un'appearance action-button che eredita da button e ridefinisce solo i suoi stati default, hovered e pressed, verifica ogni modifica nell'anteprima del Theme Builder e carica il tema nella Adaptive Operations Console.

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

- ✓ AdaptiveOps.theme creato da un tema di base e caricato dall'applicazione
- ✓ Token di colore con nome e token di font default, heading e mono definiti una volta sola e riutilizzati
- ✓ Appearance di button, panel, tab, intestazione della griglia, editor non valido e tooltip ridisegnate a partire dai token
- ✓ Appearance action-button che eredita da button con gli stati default, hovered e pressed ridefiniti
- ✓ Screenshot dell'anteprima del Theme Builder e una nota sulle decisioni relative all'ordine degli stati e all'ereditarietà

Perché è fatto così

Il Modulo 1 ha dato ad Adaptive Operations Console uno shell in dock, ma il tema di base possiede ancora ogni colore e font. La scorciatoia è brandizzarla controllo per controllo con BackColor, Font, stringhe CssStyle o CSS contro il DOM interno dei widget. Ognuna di queste strade disperde un valore esadecimale in decine di punti, combatte il motore dei temi che già visualizza ogni stato e si rompe al prossimo

aggiornamento. Un tema è un unico file JSON che possiede l'identità dei controlli standard; il Theme Builder è lo strumento con cui lo produci e lo mantieni, e questo modulo ci costruisce il tema AdaptiveOps. 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
tema di base	Il tema integrato che il Theme Builder copia per creare il tuo, così centinaia di appearance, componenti, stati, metriche, immagini e font esistono già prima che tu cambi un valore.
token con nome per colori e font	Voci nelle sezioni colors e fonts (brandPrimary, danger, heading) che le appearance richiamano per nome, così un cambio di brand è una modifica sola.
appearance e AppearanceKey	Un'appearance è il blocco del tema per un tipo di widget; l' AppearanceKey di un controllo seleziona quale blocco si applica, ed è così che esiste una variante semantica come un pulsante di azione.
ereditarietà delle appearance	Una voce inherit permette a un'appearance specializzata di riutilizzarne una di base e ridefinire solo pochi stati, così action-button resta un vero pulsante in tutto il resto.
ordine degli stati	Stati come default, hovered, pressed, focused, disabled e invalid vengono applicati nell'ordine dell'elenco; spostarne uno cambia quali stili vincono, quindi l'ordine è comportamento.
componente figlio	Una parte interna di un widget, come la caption bar di un pannello, l'intestazione di colonna di una griglia o il pulsante di un editor, con il proprio percorso di appearance e i propri stati.
stili e proprietà	Gli stili sono valori simili al CSS che il tema trasforma in classi CSS dinamiche e possiede; le proprietà sono proprietà del widget che possono diventare stile inline o comportamento e possono essere scavalcate dall'applicazione.

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 AdaptiveOps 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 - "Theme Builder e struttura interna del JSON dei temi" - del corso LearnWisej "Temi e layout responsive", 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/theming-and-responsive-layouts/specs/module-02.md>

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

1. Progetto "AdaptiveOps", 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 tema AdaptiveOps nel Theme Builder partendo da un tema di base integrato. Definisci una palette di colori con nome con `brandPrimary`, `brandAccent`, `surface`, `surfaceAlt`, `textMain`, `textMuted`, `danger`, `warning`, `success` e `focusFrame` più i font `default`, `heading` e `mono`, poi ridisegna le appearance di `button`, `panel`, `tab`, intestazione della griglia, editor non valido e tooltip usando quei token. Aggiungi un'appearance `action-button` che eredita da `button` e ridefinisce solo i suoi stati `default`, `hovered` e `pressed`, verifica ogni modifica nell'anteprima del Theme Builder e carica il tema nella Adaptive Operations Console.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - `AdaptiveOps.theme` creato da un tema di base e caricato dall'applicazione
 - Token di colore con nome e token di font `default`, `heading` e `mono` definiti una volta sola e riutilizzati
 - Appearance di `button`, `panel`, `tab`, intestazione della griglia, editor non valido e tooltip ridisegnate a partire dai token
 - Appearance `action-button` che eredita da `button` con gli stati `default`, `hovered` e `pressed` ridefiniti
 - Screenshot dell'anteprima del Theme Builder e una nota sulle decisioni relative all'ordine degli stati e all'ereditarietà
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:

- Domani il marketing cambia `brandPrimary` con un nuovo valore esadecimale. Quali file e quali righe cambiano nella tua soluzione, e quale sarebbe stata la risposta se invece avessi impostato `BackColor` su ogni pulsante?
- Hai spostato lo stato `pressed` sopra `hovered` nell'appearance `action-button`. Che cosa vede un utente mentre tiene premuto il pulsante del mouse, e perché a deciderlo è l'ordine e non il colore?
- Una `TextBox` nell'editor dei dettagli resta grigia quando non è valida, anche se hai dato stile allo stato `invalid`. Quali tre cose controlli, e in quale ordine, prima di modificare

di nuovo il tema?

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:

- ✓ Domani il marketing cambia brandPrimary con un nuovo valore esadecimale. Quali file e quali righe cambiano nella tua soluzione, e quale sarebbe stata la risposta se invece avessi impostato BackColor su ogni pulsante?
- ✓ Hai spostato lo stato pressed sopra hovered nell'appearance action-button. Che cosa vede un utente mentre tiene premuto il pulsante del mouse, e perché a deciderlo è l'ordine e non il colore?
- ✓ Una TextBox nell'editor dei dettagli resta grigia quando non è valida, anche se hai dato stile allo stato invalid. Quali tre cose controlli, e in quale ordine, prima di modificare di nuovo il tema?

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 2 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 2 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Theming and Responsive Layouts Course/Module 2`.



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.

- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/theming-and-responsive-layouts/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."*
- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice AdaptiveOps 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 AdaptiveOps 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.