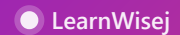


Esercizio di coding con l'IA

Costruisci il progetto VisualOperationsStudio 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 7 ti ha appena insegnato.

Che cosa stai chiedendo

Completa il capstone di VisualOperationsStudio mettendo un unico modello dietro tutte e quattro le superfici. Ristruttura il progetto in cartelle Models, Geometry, Renderers e adapter Wisej, così che l'indicatore disegnato, le celle di griglia disegnate a mano, la topologia su Canvas e l'export PNG leggano tutti lo stesso OperationsModel e condividano il livello di geometria puro, senza stato di business duplicato per renderer. Aggiungi un helper RenderMetrics che registra la durata del rendering, la dimensione del bitmap prodotto e il numero di oggetti della scena disegnati, e usalo per confrontare l'indicatore disegnato sul server con il renderer off-screen prima e dopo un'ottimizzazione che scegli e motivi. Metti alla prova i percorsi di errore — dati vuoti, superficie a dimensione zero, intervallo non valido, font mancante — così che ciascuno degradi in un segnaposto leggibile invece che in un'eccezione, aggiungi una tabella accessibile che elenca ogni valore mostrato dalla grafica e scrivi le note del capstone su architettura, confine browser/server e matrice di test.

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

- ✓ Livelli Models, Geometry, Renderers e adapter con un solo OperationsModel che guida tutte e quattro le superfici
- ✓ Helper RenderMetrics che registra durata del rendering, dimensione del bitmap e numero di oggetti disegnati per ciascuna superficie
- ✓ Misure prima e dopo per un'ottimizzazione, con la motivazione della scelta
- ✓ Percorsi di errore per dati vuoti, dimensione zero, intervallo non valido e font mancante che degradano invece di sollevare eccezioni
- ✓ CapstoneNotes.md con l'architettura, il confine server/browser, la matrice di test e una tabella di dati accessibile

Perché è fatto così

VisualOperationsStudio possiede ormai quattro superfici di rendering, e ciascuna è stata costruita nel proprio modulo con la propria idea privata di che cosa significhi la salute di una macchina. È così che un sottosistema di disegno marcisce: qualcuno sposta una soglia di warning, tre superfici sono d'accordo e il PNG esportato contraddice in silenzio la schermata da cui proviene. Questo modulo fa collassare quelle quattro interpretazioni in un unico modello dietro un unico livello di geometria, e poi dimostra con misure, percorsi di errore deliberati e un equivalente semantico che il risultato regge dati veri, piattaforme vere e utenti veri.

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 visuale	Solo stato semantico: letture, didascalie, nodi e archi, serie di andamento, gravità come scelta di dominio e selezione condivisa, così che nessuna superficie abbia bisogno di una copia.
Livello di geometria	Matematica quasi pura che trasforma lo stato del modello più la dimensione corrente della superficie in rettangoli, angoli di apertura, posizioni delle etichette e regioni di hit, testabile senza alcun controllo presente.
Adapter di superficie	Un renderer per superficie invece di un'interfaccia comune: System.Drawing.Graphics e Wisej.Web.Canvas mantengono le proprie primitive sopra una geometria condivisa.
Budget di prestazioni	Un numero concordato prima di mettere mano alle ottimizzazioni, per esempio un controllo di stato disegnato sotto i 5 ms e un redraw completo del Canvas entro circa 16-50 ms.
Strumentazione del rendering	Tempo trascorso registrato, dimensioni del bitmap, numero di oggetti disegnati e crescita del payload per ciascuna superficie, così che un'ottimizzazione si argomenti con un prima e un dopo.
Test a immagine di riferimento	Un rendering off-screen a input fisso confrontato con un riferimento memorizzato usando una tolleranza, perché antialiasing ed encoder differiscono fra piattaforme.
Equivalente semantico	L'etichetta, l' AccessibleDescription, il tooltip o la lista di accompagnamento che porta ogni valore essenziale mostrato dalla grafica, così che nulla di importante esista solo come pixel.

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 7. Puoi aprire tu stesso [module-07.md](#) per vedere esattamente che cosa leggerà.



Usa ChatGPT



Usa Claude

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.

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 VisualOperationsStudio 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 7 - "Architettura di disegno in produzione, prestazioni e capstone" - del corso LearnWisej "Disegno e grafica in Wisej.NET", 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/drawing-and-painting/specs/module-07.md>

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

1. Progetto "VisualOperationsStudio", 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: Completa il capstone di VisualOperationsStudio mettendo un unico modello dietro tutte e quattro le superfici. Ristruttura il progetto in cartelle Models, Geometry, Renderers e adapter Wisej, così che l'indicatore disegnato, le celle di griglia disegnate a mano, la topologia su Canvas e l'export PNG leggano tutti lo stesso OperationsModel e condividano il livello di geometria puro, senza stato di business duplicato per renderer. Aggiungi un helper RenderMetrics che registra la durata del rendering, la dimensione del bitmap prodotto e il numero di oggetti della scena disegnati, e usalo per confrontare l'indicatore disegnato sul server con il renderer off-screen prima e dopo un'ottimizzazione che scegli e motivi. Metti alla prova i percorsi di errore - dati vuoti, superficie a dimensione zero, intervallo non valido, font mancante - così che ciascuno degradi in un segnaposto leggibile invece che in un'eccezione, aggiungi una tabella accessibile che elenca ogni valore mostrato dalla grafica e scrivi le note del capstone su architettura, confine browser/server e matrice di test.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Livelli Models, Geometry, Renderers e adapter con un solo OperationsModel che guida tutte e quattro le superfici
 - Helper RenderMetrics che registra durata del rendering, dimensione del bitmap e numero di oggetti disegnati per ciascuna superficie
 - Misure prima e dopo per un'ottimizzazione, con la motivazione della scelta
 - Percorsi di errore per dati vuoti, dimensione zero, intervallo non valido e font mancante che degradano invece di sollevare eccezioni
 - CapstoneNotes.md con l'architettura, il confine server/browser, la matrice di test e una tabella di dati accessibile
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:

- Il tuo livello di geometria riceve il modello e una dimensione e restituisce forme, ma un membro del team vorrebbe che restituisse un GraphicsPath così che il renderer dell'indicatore si accorci. Che cosa costa questo al renderer Canvas e agli unit test, e

dove tracteresti invece il confine?

- Il PNG esportato e il Canvas a schermo differiscono di qualche pixel sull'agente di build Linux ma coincidono su Windows. Quali misure e quale tipo di test ti direbbero se si tratta di un difetto o di una differenza di piattaforma attesa?
- Chi rivede il tuo lavoro dice che la tabella accessibile è ridondante perché ogni indicatore ha già un tooltip. Spiega che cosa dà la tabella a chi usa solo la tastiera o uno screen reader e che il tooltip non dà, e che cosa implica per i chip di gravità della griglia basati solo sul colore.

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:

- ✓ Il tuo livello di geometria riceve il modello e una dimensione e restituisce forme, ma un membro del team vorrebbe che restituisse un GraphicsPath così che il renderer dell'indicatore si accorci. Che cosa costa questo al renderer Canvas e agli unit test, e dove tracteresti invece il confine?
- ✓ Il PNG esportato e il Canvas a schermo differiscono di qualche pixel sull'agente di build Linux ma coincidono su Windows. Quali misure e quale tipo di test ti direbbero se si tratta di un difetto o di una differenza di piattaforma attesa?
- ✓ Chi rivede il tuo lavoro dice che la tabella accessibile è ridondante perché ogni indicatore ha già un tooltip. Spiega che cosa dà la tabella a chi usa solo la tastiera o uno screen reader e che il tooltip non dà, e che cosa implica per i chip di gravità della griglia basati solo sul colore.

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*

painting/specs/module-07.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 VisualOperationsStudio 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 VisualOperationsStudio 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.