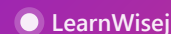


Esercizio di coding con l'IA

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.



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

Aggiungi a VisualOperationsStudio un TopologyImageRenderer che trasforma il modello di topologia in un PNG senza toccare un solo controllo Wisej. Dagli un metodo Render che riceve la scena e una dimensione di output, crea il Bitmap, ne ottiene il Graphics con Graphics.FromImage, disegna nodi, archi, etichette e una legenda con pen, brush e un GraphicsPath, e fa il dispose di tutto ciò che possiede lasciando in pace gli oggetti del chiamante. Risolvi il suo Font attraverso un metodo ResolveFont che prova una famiglia fornita dall'applicazione e ripiega su una sicuramente presente, invece di dare per scontato che un font di sistema esista; salva in un MemoryStream come PNG e passa i byte alla pagina per il download. Poi controlla i riferimenti del progetto per confermare che non stai aggiungendo System.Drawing.Common accanto all'implementazione managed di Wisej, esegui o prepara il renderer su Windows e in un container Linux, e metti per iscritto come si comporta lo stesso codice sui target iOS, Android e macOS Hybrid.

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

- ✓ TopologyImageRenderer senza alcuna dipendenza da un controllo Wisej, che rende tramite Graphics.FromImage dentro un Bitmap
- ✓ Metodo ResolveFont con una catena di fallback esplicita invece di un font di sistema dato per scontato
- ✓ PNG scritto in un MemoryStream e scaricato dalla pagina, con il dispose di ogni oggetto grafico posseduto
- ✓ Verifica dei riferimenti che confermi l'uso di Managed.System.Drawing e l'assenza di System.Drawing.Common affiancato
- ✓ CrossPlatform.md con le prove su Windows e in container Linux e le implicazioni per iOS, Android e macOS

Perché è fatto così

Tutto quello che VisualOperationsStudio ha disegnato finora appartiene a una sessione viva: l'indicatore si ridisegna perché un controllo l'ha chiesto, e la topologia sul Canvas del Modulo 5 esiste soltanto come comandi che il browser ha già eseguito. Ora operations vuole quella stessa topologia dentro una e-mail notturna sugli incidenti e allegata a un ticket, il che significa un'immagine senza alcun controllo dietro. Nel frattempo l'applicazione si sta spostando in un container Linux, quindi il codice di disegno deve continuare a funzionare su un host senza GDI+ e senza libgdiplus.

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
System.Drawing.Common	Il wrapper di GDI+ di Microsoft, specifico per Windows a partire da .NET 6; l'interruttore per Unix è sparito in .NET 7, quindi non è un'opzione per Linux, macOS, iOS o Android.
Managed.System.Drawing	L'id del package NuGet, invertito perché la radice System.* dei package è riservata; Wisej.NET 4 lo riferenzia già al posto tuo.
System.Drawing.Managed.dll	L'assembly consegnato da quel package — l'implementazione managed vera e propria, che vedi in un log di build più che nel tuo sorgente.
Il namespace System.Drawing	Invariato: Bitmap, Graphics, Pen e Font mantengono i loro nomi, quindi il codice di disegno esistente compila senza riscritture.
Motore grafico managed	Un'implementazione puramente managed su un motore ImageSharp personalizzato e integrato; sulle versioni moderne di .NET evita GDI+, libgdiplus e le chiamate native.
Graphics.FromImage	Il punto di ingresso off-screen: una superficie di disegno sopra un Bitmap che possiedi, senza controllo, senza evento Paint e senza sessione.
PrivateFontCollection	Carica i file di font distribuiti con l'applicazione, così l'output dipende da ciò che hai rilasciato e non da ciò che capita sia installato sull'host.

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 6. Puoi aprire tu stesso [module-06.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



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece

Codex; qualsiasi piano che possa navigare il web recupererà la specifica.

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 6 - "System.Drawing.Managed su Windows, Linux, macOS, iOS e Android" - 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-06.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: Aggiungi a VisualOperationsStudio un TopologyImageRenderer che trasforma il modello di topologia in un PNG senza toccare un solo controllo Wisej. Dagli un metodo Render che riceve la scena e una dimensione di output, crea il Bitmap, ne ottiene il Graphics con Graphics.FromImage, disegna nodi, archi, etichette e una legenda con pen, brush e un GraphicsPath, e fa il dispose di tutto ciò che possiede lasciando in pace gli oggetti del chiamante. Risolvi il suo Font attraverso un metodo ResolveFont che prova una famiglia fornita dall'applicazione e ripiega su una sicuramente presente, invece di dare per scontato che un font di sistema esista; salva in un MemoryStream come PNG e passa i byte alla pagina per il download. Poi controlla i riferimenti del progetto per confermare che non stai aggiungendo System.Drawing.Common accanto all'implementazione managed di Wisej, esegui o prepara il renderer su Windows e in un container Linux, e metti per iscritto come si comporta lo stesso codice sui target iOS, Android e macOS Hybrid.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - TopologyImageRenderer senza alcuna dipendenza da un controllo Wisej, che rende tramite Graphics.FromImage dentro un Bitmap
 - Metodo ResolveFont con una catena di fallback esplicita invece di un font di sistema dato per scontato
 - PNG scritto in un MemoryStream e scaricato dalla pagina, con il dispose di ogni oggetto grafico posseduto
 - Verifica dei riferimenti che conferma l'uso di Managed.System.Drawing e l'assenza di System.Drawing.Common affiancato
 - CrossPlatform.md con le prove su Windows e in container Linux e le implicazioni per iOS, Android e macOS
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 collega sostiene che il renderer dimostra che l'applicazione è multiplatforma perché il codice compila su Linux. Che cosa dovresti osservare davvero nel container prima di

accettare quell'affermazione, e quale parte dell'output è più probabile che differisca?

- Il tuo renderer fa il dispose del Bitmap che ha creato. Perché fare il dispose del Graphics dentro l'handler di Paint di un controllo sarebbe un bug, e quale regola copre entrambi i casi in una sola frase?
- Qualcuno propone di sostituire questo renderer con SkiaSharp perché nei benchmark è più veloce. Che cosa dovresti sapere del carico di lavoro prima di dire di sì, e che cosa perde il team abbandonando l'API System.Drawing?

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 collega sostiene che il renderer dimostra che l'applicazione è multiplatforma perché il codice compila su Linux. Che cosa dovresti osservare davvero nel container prima di accettare quell'affermazione, e quale parte dell'output è più probabile che differisca?
- ✓ Il tuo renderer fa il dispose del Bitmap che ha creato. Perché fare il dispose del Graphics dentro l'handler di Paint di un controllo sarebbe un bug, e quale regola copre entrambi i casi in una sola frase?
- ✓ Qualcuno propone di sostituire questo renderer con SkiaSharp perché nei benchmark è più veloce. Che cosa dovresti sapere del carico di lavoro prima di dire di sì, e che cosa perde il team abbandonando l'API System.Drawing?

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.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto VisualOperationsStudio 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/drawing-and-painting/specs/module-06.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."*



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.