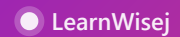


Esercizio di coding con l'IA

Costruisci il progetto AccessOps 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.



Istruiscilo, leggilo, poi mettilo alla prova.

Questo esercizio ha tre parti, e solo la prima riguarda il prompt. Istruisci un assistente con la specifica del modulo e ti fai restituire una soluzione AccessOps; leggi quello che ha scritto e decidi se fa davvero ciò che il modulo chiedeva; poi lo spingi più in là di quanto il modulo chiedesse. La seconda e la terza parte sono dove si impara.

Che cosa stai chiedendo

Aggiungi l'autorizzazione ad AccessOps: un `PermissionService` che al login risolve ruoli e claim in permessi, una `guard Authorize` richiamata da ogni metodo di `DocumentService` e `ApprovalService`, una regola a livello di record per cui i dipendenti vedono solo il proprio reparto e i manager solo i propri collaboratori diretti, una UI che nasconde le azioni di amministrazione e disabilita i pulsanti di approvazione che l'utente non può usare, e un test che dimostra che una richiesta contraffatta da parte di un non-manager viene respinta dal servizio.

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

- ✓ `PermissionService` che risolve i permessi nel `SessionContext`
- ✓ Guard di autorizzazione in ogni metodo di servizio
- ✓ Regole a livello di record per reparto e manager
- ✓ Adattamento della UI in base ai permessi
- ✓ Test negativo con una richiesta contraffatta

Perché è fatto così

Ora AccessOps sa chi ha effettuato l'accesso: il Modulo 2 ha lasciato un'identità verificata nel `SessionContext`. Sapere chi è qualcuno però non dice nulla su che cosa può fare con un documento riservato o con una richiesta di approvazione da centinaia di migliaia di euro. In un portale interno il guasto tipico raramente è un login rotto — è un addetto alle paghe che può leggere i contratti dell'ufficio legale perché la query non è mai stata limitata, o un team lead che approva la propria nota spese perché l'unica verifica era un pulsante disabilitato.

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
ruolo	Una funzione lavorativa generica con un nome, come Employee, Manager o Auditor. I ruoli sono comodi da amministrare ma troppo grossolani per verificarli direttamente, perché una schermata ha bisogno di sapere di un'azione, non di un intero mestiere.
claim	Un fatto nome/valore sull'identità verificata — reparto 14, centro di costo EU-North, manager del team 7. I claim portano i dati di cui hanno bisogno le regole a livello di record, e arrivano dall'archivio delle identità, mai dal browser.
permesso	Una singola azione con un nome che l'applicazione può eseguire, come ApproveRequest o ViewDepartmentDocuments. Il codice verifica i permessi; gli amministratori assegnano i ruoli; la mappatura tra i due è configurazione.
risoluzione dei permessi	L'unico passaggio al login che espande ruoli e claim in un insieme piatto e immutabile di permessi tenuto nel SessionContext, così nessuna richiesta rilegge la tabella dei ruoli e nessuna coppia di verifiche può dare risposte diverse.
guard di autorizzazione	Una chiamata come Require(Permission.ApproveRequest) all'inizio di un metodo di servizio, che lancia un'eccezione prima che venga fatto qualsiasi lavoro. È l'unico punto in cui l'applicazione delle regole avviene davvero.
regola a livello di record	Una restrizione su quali righe un'azione consentita può toccare — il tuo reparto, i tuoi collaboratori diretti — applicata dentro la query o il comando invece che dopo aver già caricato i dati.
adattamento della UI	Nascondere, disabilitare o spiegare un'azione in base allo stesso insieme di permessi che il server fa rispettare. È una cortesia verso l'utente e mai un controllo.

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 3. Puoi aprire tu stesso [module-03.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 AccessOps 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 3 - "Autorizzazione: ruoli, claim e permessi" - del corso LearnWisej "Pattern di sicurezza e autenticazione", 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/security-and-auth-patterns/specs/module-03.md>

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

1. Progetto "AccessOps", 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 l'autorizzazione ad AccessOps: un `PermissionService` che al login risolve ruoli e claim in permessi, una `guard Authorize` richiamata da ogni metodo di `DocumentService` e `ApprovalService`, una regola a livello di record per cui i dipendenti vedono solo il proprio reparto e i manager solo i propri collaboratori diretti, una UI che nasconde le azioni di amministrazione e disabilita i pulsanti di approvazione che l'utente non può usare, e un test che dimostra che una richiesta contraffatta da parte di un non-manager viene respinta dal servizio.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - `PermissionService` che risolve i permessi nel `SessionContext`
 - `Guard` di autorizzazione in ogni metodo di servizio
 - Regole a livello di record per reparto e manager
 - Adattamento della UI in base ai permessi
 - Test negativo con una richiesta contraffatta
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:

- A un auditor vengono concessi i diritti di approvazione mentre la sua sessione AccessOps è ancora aperta. Spiega esattamente quando il nuovo permesso ha effetto con il tuo design, perché hai scelto quel momento e che cosa dovresti cambiare per renderlo immediato.
- La tua query dei documenti filtra per reparto nel repository. Descrivi che cosa guadagna un attaccante se quel filtro viene spostato invece sul DataSource della griglia, anche se il risultato visibile a schermo è identico.
- Sia `ApprovalService` sia il suo repository potrebbero verificare che l'approvatore gestisca il richiedente. Argomenta quale dei due sia il posto giusto, e di' che cosa si rompe se la verifica sta nell'altro, o in entrambi.

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:

- ✓ A un auditor vengono concessi i diritti di approvazione mentre la sua sessione AccessOps è ancora aperta. Spiega esattamente quando il nuovo permesso ha effetto con il tuo design, perché hai scelto quel momento e che cosa dovresti cambiare per renderlo immediato.
- ✓ La tua query dei documenti filtra per reparto nel repository. Descrivi che cosa guadagna un attaccante se quel filtro viene spostato invece sul DataSource della griglia, anche se il risultato visibile a schermo è identico.
- ✓ Sia ApprovalService sia il suo repository potrebbero verificare che l'approvatore gestisca il richiedente. Argomenta quale dei due sia il posto giusto, e di' che cosa si rompe se la verifica sta nell'altro, o in entrambi.

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 AccessOps 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 AccessOps 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 AccessOps 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.