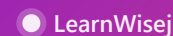


Esercizio di coding con l'IA

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.



Scrivere il codice è ormai la metà facile.

Ottenere codice funzionante per "Integrazione con OAuth e OpenID Connect" da un assistente moderno richiede un buon prompt. Giudicarlo, correggerlo ed estenderlo richiede tutto quello che questo modulo ha trattato. È su questa divisione che è costruito l'esercizio: tu metti la specifica e la revisione, l'assistente mette la digitazione.

Che cosa stai chiedendo

Aggiungi l'accesso esterno ad AccessOps: un client OpenID Connect che usa l'authorization code flow con PKCE verso un identity provider di test, un gestore di callback che valida l'ID token e il suo nonce, una mappatura dei claim che popola il SessionContext e crea un record dipendente locale al primo accesso, un login locale di riserva per gli account di servizio, e una disconnessione che chiude sia la sessione locale sia quella del provider.

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

- ✓ Configurazione del client OIDC con PKCE
- ✓ Gestione del callback con validazione di token e nonce
- ✓ Mappatura dei claim e provisioning al primo accesso
- ✓ Login locale di riserva per gli account di servizio
- ✓ Disconnessione che copre entrambe le sessioni

Perché è fatto così

AccessOps sa già verificare una password tutta sua, ma l'organizzazione che lo usa non vuole una seconda directory: il personale accede a tutto il resto con l'identity provider aziendale, e ingressi, uscite e policy di autenticazione a più fattori si amministrano in un solo posto. Delegare il login toglie del tutto la password da AccessOps e la sostituisce con un obbligo diverso — ora il portale riceve un'identità da una terza parte, attraverso il browser dell'utente stesso, e deve dimostrare che quell'identità è autentica, non modificata e destinata proprio a lui prima di mostrare un solo record di reparto.

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
authorization code flow	Il flusso OpenID Connect in cui il browser trasporta soltanto un code monouso di breve durata, mentre i token vengono recuperati dal tuo server con una chiamata HTTPS diretta che il browser non vede mai.
PKCE	Un code verifier casuale generato per ogni accesso, inviato all'endpoint authorize solo come hash SHA-256 e rivelato all'endpoint token, così un code intercettato non può essere riscattato da nessun altro.
ID token	Un JWT firmato che afferma che questo provider ha autenticato questo subject, per questo client, in questo momento. È l'unico token che AccessOps consulta per decidere chi è l'utente.
access token e refresh token	L'access token è una credenziale bearer per chiamare l'API di qualcun altro; il refresh token ottiene dei sostituti nel back channel. Nessuno dei due è una prova d'identità per la tua app.
state e nonce	Due valori casuali che generi a ogni accesso: state lega il callback alla richiesta che hai davvero avviato, nonce lega l'ID token restituito a quella stessa richiesta così che un token più vecchio non possa esservi riprodotto.
redirect URI	L'URL esatto, registrato in anticipo, a cui il provider rimanda il browser. In AccessOps è un piccolo endpoint lato server e non una finestra Wisej.NET, perché deve essere eseguito prima che esista qualsiasi UI.
mappatura dei claim e provisioning	Trasformare i claim del provider in un dipendente di AccessOps: la coppia issuer e subject identifica l'account, e al primo accesso viene creata una riga locale senza alcun permesso finché qualcuno non li concede.

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 4. Puoi aprire tu stesso [module-04.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 4 - "Integrazione con OAuth e OpenID Connect" - 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-04.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'accesso esterno ad AccessOps: un client OpenID Connect che usa l'authorization code flow con PKCE verso un identity provider di test, un gestore di callback che valida l'ID token e il suo nonce, una mappatura dei claim che popola il SessionContext e crea un record dipendente locale al primo accesso, un login locale di riserva per gli account di servizio, e una disconnessione che chiude sia la sessione locale sia quella del provider.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Configurazione del client OIDC con PKCE
 - Gestione del callback con validazione di token e nonce
 - Mappatura dei claim e provisioning al primo accesso
 - Login locale di riserva per gli account di servizio
 - Disconnessione che copre entrambe le sessioni
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 attaccante recupera l'authorization code dalla cronologia del browser su una macchina condivisa. Che cosa gli impedisce di scambiarlo con i token, e che cosa non sarebbe riuscito a fermarlo senza PKCE?
- Il tuo provider emette claim di gruppo identici ai nomi dei permessi di AccessOps. Perché il PermissionService risolve comunque i permessi da sé invece di copiare quei claim direttamente nel SessionContext?
- Un utente clicca Sign out, non chiude nulla e clicca subito di nuovo Sign in - e si ritrova dentro AccessOps senza digitare niente. Quale sessione è sopravvissuta, e quale redirect mancava?

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 attaccante recupera l'authorization code dalla cronologia del browser su una macchina condivisa. Che cosa gli impedisce di scambiarlo con i token, e che cosa non sarebbe riuscito a fermarlo senza PKCE?
- ✓ Il tuo provider emette claim di gruppo identici ai nomi dei permessi di AccessOps. Perché il PermissionService risolve comunque i permessi da sé invece di copiare quei claim direttamente nel SessionContext?
- ✓ Un utente clicca Sign out, non chiude nulla e clicca subito di nuovo Sign in — e si ritrova dentro AccessOps senza digitare niente. Quale sessione è sopravvissuta, e quale redirect mancava?

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.

- ✓ **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."*

- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-04.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."*



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.