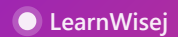


Esercizio di coding con l'IA

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.



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

Implementa l'autenticazione di AccessOps: una pagina di login con blocco dell'account dopo cinque tentativi falliti, la verifica della password con un hash moderno con salt, un SessionContext creato solo dopo la verifica e memorizzato nella sessione, un timeout di inattività che riporta l'utente al login e ne ripulisce lo stato, un Logout esplicito che chiude la sessione e un record di audit per ogni tentativo di accesso.

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

- ✓ Pagina di login con blocco dell'account e messaggi di errore generici
- ✓ Servizio di hashing e verifica delle password
- ✓ SessionContext con identità verificata
- ✓ Timeout di inattività e logout esplicito
- ✓ Record di audit dei login

Perché è fatto così

AccessOps contiene anagrafiche dei dipendenti, documenti riservati e richieste di approvazione, quindi la prima schermata che chiunque raggiunge è anche il primo controllo che conta. Tutto ciò che il resto di questo corso fa rispettare — permessi, regole a livello di record, audit trail — poggia su un unico presupposto: che il server sappia chi c'è dall'altra parte di questa sessione e sia in grado di dimostrarlo. Il Modulo 1 ha disegnato i confini di fiducia; questo modulo mette un'identità verificata dentro il confine di sessione e impedisce che venga indovinata, presa in prestito o fatta sopravvivere oltre il dovuto.

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
hash lento con salt	Una derivazione a senso unico della password con un salt casuale per utente e un numero di iterazioni volutamente costoso, così che una tabella rubata non possa essere invertita e ogni tentativo costi lavoro vero.
confronto a tempo costante	Confrontare due hash con un numero fisso di operazioni usando <code>CryptographicOperations.FixedTimeEquals</code> , così il tempo di risposta del server non rivela mai quanti byte di un tentativo erano giusti.
SessionContext	L'oggetto creato solo dopo la verifica di una password, che contiene id del dipendente, reparto, ruoli e dati di accesso, ed è memorizzato in <code>Application.Session</code> come unico punto a cui qualsiasi codice chiede chi è l'utente.
timeout di sessione	Il periodo di inattività dopo il quale Wisej.NET elimina la sessione lato server e tutto ciò che contiene, configurato per l'applicazione in <code>Default.json</code> invece che stabilito a occhio schermata per schermata.
session fixation	Un attacco in cui l'identificatore che ha portato la pagina di login anonima porta anche, dopo, la sessione autenticata, così che un identificatore piazzato dall'attaccante diventa un identificatore con accesso effettuato.
throttling del login	Contare i fallimenti per account e per indirizzo del client e rifiutare ulteriori tentativi per un periodo di blocco: è questo che trasforma una macchina per tirare a indovinare senza limiti in pochi tentativi all'ora.
record di audit del login	Una riga per tentativo con il nome utente provato, l'ora, l'indirizzo del client e l'esito reale — il posto in cui viene scritto il motivo specifico del fallimento, dato che lo schermo ne mostra sempre solo uno generico.

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 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 2 - "Autenticazione e identità di sessione" - 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-02.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: Implementa l'autenticazione di AccessOps: una pagina di login con blocco dell'account dopo cinque tentativi falliti, la verifica della password con un hash moderno con salt, un `SessionContext` creato solo dopo la verifica e memorizzato nella sessione, un timeout di inattività che riporta l'utente al login e ne ripulisce lo stato, un Logout esplicito che chiude la sessione e un record di audit per ogni tentativo di accesso.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Pagina di login con blocco dell'account e messaggi di errore generici
 - Servizio di hashing e verifica delle password
 - `SessionContext` con identità verificata
 - Timeout di inattività e logout esplicito
 - Record di audit dei login
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:

- La tua pagina di login risponde a una password errata in 480 ms e a un nome utente sconosciuto in 6 ms. Spiega che cosa impara un attaccante da questa sola differenza, e descrivi le due modifiche che eliminano il segnale senza cambiare il messaggio.
- Un collega propone di tenere il dipendente autenticato in una proprietà statica perché "è più semplice che andare a pescare nella sessione ogni volta". Descrivi esattamente che cosa vede il secondo utente contemporaneo, e in quale momento le due sessioni interferiscono per la prima volta.
- Il tuo pulsante Logout ripulisce il `SessionContext` ma non chiude la sessione. Indica due cose che restano recuperabili dopo, e di' quale consideri più grave in un portale che mostra documenti riservati.

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:

- ✓ La tua pagina di login risponde a una password errata in 480 ms e a un nome utente sconosciuto in 6 ms. Spiega che cosa impara un attaccante da questa sola differenza, e descrivi le due modifiche che eliminano il segnale senza cambiare il messaggio.
- ✓ Un collega propone di tenere il dipendente autenticato in una proprietà statica perché "è più semplice che andare a pescare nella sessione ogni volta". Descrivi esattamente che cosa vede il secondo utente contemporaneo, e in quale momento le due sessioni interferiscono per la prima volta.
- ✓ Il tuo pulsante Logout ripulisce il SessionContext ma non chiude la sessione. Indica due cose che restano recuperabili dopo, e di' quale consideri più grave in un portale che mostra documenti riservati.

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.

- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice AccessOps 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 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."*



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.