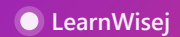


Esercizio di coding con l'IA

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.



Lo stesso modulo, costruito come costruirai davvero le cose.

Tutte le altre sezioni di questo modulo spiegano come funziona "Proteggere dati, upload, download e handler". Questa ti chiede di produrlo: dirigendo un assistente con la specifica del modulo stesso, poi verificando che il risultato giri e faccia quello che descrive la traccia del lab. È il flusso di lavoro che userai sul lavoro vero, provato su qualcosa di cui conosci già la risposta giusta.

Che cosa stai chiedendo

Metti in sicurezza la funzione documenti di AccessOps: upload limitati per dimensione e tipo con content sniffing, salvati fuori dalla web root con un nome casuale e collegati al reparto proprietario; download che passano dall'autorizzazione di DocumentService e da Application.Download; un handler personalizzato per un link di condivisione firmato che scade dopo un'ora; e una revisione che verifica che ogni query sia parametrizzata e che ogni stringa fornita dall'utente sia codificata prima di essere mostrata.

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

- ✓ Pipeline di upload validata con cartella di quarantena
- ✓ Download autorizzato tramite il servizio
- ✓ Handler per link di condivisione firmati con scadenza
- ✓ Revisione di query parametrizzate e codifica dell'output
- ✓ Test con una richiesta di download contraffatta

Perché è fatto così

L'area dei documenti riservati è la parte di AccessOps che un attaccante vuole davvero: lettere sugli stipendi, fascicoli disciplinari, contratti firmati, documentazione del consiglio di amministrazione. Ogni altra schermata sposta righe tra griglie; questa sposta byte attraverso il confine verso un browser che non controlli, e ritorno. Identità e permessi sono stati sistemati nei moduli precedenti, e questo modulo riguarda i punti in cui un file, un URL o una stringa riescono ad aggirarli.

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
download inviato dal server	Chiamare Application.Download dal codice server invia un file al browser durante un evento che hai già autorizzato, quindi non ha un indirizzo da salvare, condividere o indovinare.
esposizione dei file statici	Tutto ciò che sta sotto la web root viene servito prima che giri una sola riga del tuo codice, quindi i permessi del file sono esattamente "chiunque conosca l'URL".
evento Uploaded	Il controllo Upload lo solleva sul server con gli stream dei file in arrivo — l'unico punto in cui un file può essere ispezionato prima di entrare nel tuo sistema.
verifica della firma del contenuto	Confrontare i primi byte di un file con l'intestazione nota del formato, perché l'estensione e il content type dichiarato sono forniti dal client.
archiviazione in quarantena	Tenere l'upload accettato fuori dalla web root con un nome generato finché la scansione non dà esito pulito, e solo allora collegarlo al reparto proprietario.
handler HTTP personalizzato	Una classe server registrata nell'applicazione che riceve richieste grezze fuori dal flusso degli eventi, quindi non ha un'identità di sessione e deve autenticare e validare tutto da sé.
link firmato con scadenza	Un URL che porta un id di documento, una scadenza e un HMAC calcolato su entrambi con una chiave tenuta sul server, così i valori non possono essere modificati e il link smette di funzionare da solo.

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 5. Puoi aprire tu stesso [module-05.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 5 - "Proteggere dati, upload, download e handler" - 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-05.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: Metti in sicurezza la funzione documenti di AccessOps: upload limitati per dimensione e tipo con content sniffing, salvati fuori dalla web root con un nome casuale e collegati al reparto proprietario; download che passano dall'autorizzazione di DocumentService e da Application.Download; un handler personalizzato per un link di condivisione firmato che scade dopo un'ora; e una revisione che verifica che ogni query sia parametrizzata e che ogni stringa fornita dall'utente sia codificata prima di essere mostrata.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Pipeline di upload validata con cartella di quarantena
 - Download autorizzato tramite il servizio
 - Handler per link di condivisione firmati con scadenza
 - Revisione di query parametrizzate e codifica dell'output
 - Test con una richiesta di download 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:

- Il tuo gestore di upload verifica l'estensione rispetto a una allow-list e il file passa. Spiega che cosa può ancora inviare un attaccante, e che cosa aggiunge la verifica della firma che la verifica dell'estensione non può dare.
- Un manager inoltra un link di condivisione a qualcuno fuori dall'azienda prima che scada. Quali parti del tuo design limitano il danno, e che cosa dovresti aggiungere per revocare quel singolo link senza cambiare la chiave per tutti?
- Il tuo pulsante di download è nascosto agli utenti senza permesso di lettura, e il test con la richiesta contraffatta passa comunque. Spiega con precisione quale riga di codice lo fa passare, e perché il pulsante nascosto non c'entra nulla.

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 gestore di upload verifica l'estensione rispetto a una allow-list e il file passa. Spiega che cosa può ancora inviare un attaccante, e che cosa aggiunge la verifica della firma che la verifica dell'estensione non può dare.
- ✓ Un manager inoltra un link di condivisione a qualcuno fuori dall'azienda prima che scada. Quali parti del tuo design limitano il danno, e che cosa dovresti aggiungere per revocare quel singolo link senza cambiare la chiave per tutti?
- ✓ Il tuo pulsante di download è nascosto agli utenti senza permesso di lettura, e il test con la richiesta contraffatta passa comunque. Spiega con precisione quale riga di codice lo fa passare, e perché il pulsante nascosto non c'entra nulla.

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 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-05.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 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."*



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.