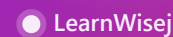


Sviluppo assistito dall'IA, documentazione pronta per MCP e consegna del capstone

Guida della lezione · Guida al lab / esame



Guida della lezione

Disciplina nel coding assistito dall'IA, accesso alla documentazione, prompt di revisione, difesa enterprise finale. Che cosa significa per un team Wisej.NET avanzato — e come affrontare questo modulo.



Focus di produzione

Disciplina nel coding assistito dall'IA, accesso alla documentazione, prompt di revisione, difesa enterprise finale.

1. Il problema in produzione

I team avanzati useranno strumenti di IA, ma il lavoro enterprise con Wisej.NET richiede comunque architettura, documentazione basata sulle fonti, code review, revisione della sicurezza e responsabilità operativa.

2. Concetti chiave

Concetto	Perché è importante
sviluppo assistito dall'IA	Usare strumenti di IA per abbozzare codice, test e documentazione, mentre il team resta responsabile di rivedere, verificare sulle fonti e fare proprio il risultato.
prompt basati sulle fonti	Prompt che indirizzano l'assistente al codice, alla documentazione e alle convenzioni reali del progetto, così l'output generato corrisponde all'architettura vera.
endpoint MCP della documentazione	Un server Model Context Protocol che espone la documentazione del team agli strumenti di IA, così le risposte arrivano dalle fonti del progetto stesso.
checklist di revisione	Un elenco fisso di verifiche su architettura, sicurezza e convenzioni applicato a ogni modifica generata dall'IA prima che venga accettata.
difesa dell'architettura	Presentare e giustificare con evidenze le decisioni progettuali del capstone, così il team può metterle in discussione e registrarne l'esito.
demo del capstone	La dimostrazione end-to-end dell'applicazione finita, che prova che i pattern dei moduli funzionano insieme in condizioni realistiche.

3. Il filo della lezione

L'IA aiuta il team a muoversi più velocemente

L'IA può abbozzare schermate, servizi, test, prompt, documentazione e inventari di migrazione. Non può farsi carico del rischio in produzione. Lo sviluppatore avanzato deve fornire il contesto architetturale, verificare le API, rivedere il codice generato e testare i percorsi di errore.

Usa la documentazione come fonte per gli strumenti

La documentazione di Wisej.NET può essere trattata come fonte di verità sul comportamento attuale della piattaforma. I prompt per l'IA dovrebbero chiedere codice che segua le convenzioni del progetto e dovrebbero richiedere citazioni o riferimenti quando si usano API specifiche della piattaforma.

Velocità senza codice non revisionato

L'assistenza dell'IA è più preziosa ai margini del lavoro: abbozzare lo scheletro di un servizio a partire dalla definizione di un comando, generare casi di test per una matrice dei percorsi di errore, scrivere un inventario di migrazione a partire da un file di soluzione o trasformare un runbook in una checklist. In ciascun caso lo sviluppatore sa già che aspetto ha il risultato corretto e può verificare l'output rapidamente. È meno preziosa dove la verifica è difficile, come nella logica di sicurezza, nella concorrenza o nelle API di piattaforma che lo sviluppatore non conosce bene.

La regola che il lab impone è che il codice generato è una bozza di un contributore sconosciuto. Passa attraverso la stessa revisione di qualsiasi pull request, con le stesse domande: dove risiede lo stato, che cosa può fallire, che cosa è sicuro e che cosa cambia in produzione. Il percorso di errore del lab mostra perché. Un servizio generato inventa un'API di Wisej.NET che non esiste e memorizza dati per utente in un campo statico. Entrambe le cose compilerebbero in una demo veloce ed entrambe fallirebbero con sessioni reali.

Accetta un suggerimento solo quando qualcuno del team sa spiegarlo senza lo strumento a disposizione.

Prompt e checklist basati sul progetto

La qualità del codice generato segue la qualità del contesto fornito. Ogni sessione con l'IA nel lab comincia con un'intestazione del prompt che dichiara le regole del progetto: la struttura della soluzione, dove vive la logica di business, che i campi statici non contengono mai dati di sessione, che i servizi impongono i permessi e che si possono usare solo API di Wisej.NET documentate, con un riferimento per ciascuna. La libreria di prompt raccoglie queste intestazioni e i prompt specifici per attività, così tutto il team parte dalla stessa base.

La checklist di revisione del codice generato è l'altra metà. È abbastanza breve da applicarla a ogni modifica e abbastanza specifica da intercettare i fallimenti che il corso ha insegnato: un campo statico che contiene stato dell'utente o del tenant, un gestore della UI che contiene logica di persistenza, un'API che non si trova nella documentazione, un controllo dei permessi mancante in un servizio, un'eccezione mostrata all'utente e una risorsa che non viene mai rilasciata.

- Il prompt nomina le convenzioni che l'output deve seguire?
- Ogni affermazione specifica della piattaforma nell'output cita la documentazione?
- La checklist è stata applicata e firmata da un revisore, non dall'autore?

Documentazione leggibile da strumenti e manutentori

Un progetto che verrà mantenuto con l'assistenza dell'IA ha bisogno di documentazione che uno strumento possa recuperare, non solo che una persona possa sfogliare. Un endpoint MCP della documentazione espone all'assistente le note di architettura, i decision record, le convenzioni e i runbook del team, così le risposte su questo progetto arrivano dalle fonti di questo progetto anziché da dati di addestramento generici. Lo stesso endpoint serve un nuovo membro del team che fa le stesse domande.

Funziona solo se la documentazione è strutturata. L'indice della documentazione del lab elenca ogni documento con uno scopo di una riga, le decisioni che registra e la data dell'ultima verifica. Gli architecture decision record del Modulo 1 sono le voci più preziose, perché spiegano perché esiste un certo design e uno strumento può citarli quando gli viene chiesto di modificarlo. Seguono i runbook, la matrice dei permessi, la matrice dei percorsi di errore e la tabella della configurazione.

Scrivi per il lettore che non ha mai conosciuto il team. Se una nota si basa su conoscenze che esistono solo nella testa di qualcuno, tra un anno sarà sbagliata.

Difendere il capstone come una revisione dell'architettura

Il capstone viene consegnato come una revisione dell'architettura di produzione anziché come una demo delle funzionalità. Lo script della demo accompagna un revisore attraverso EnterpriseOps Command Center in un ordine fisso: la struttura della soluzione e i decision record, una sessione consapevole del tenant e una richiesta tra tenant diversi rifiutata, un comando che passa attraverso un servizio fino al database e ritorno con la sua voce di audit, un'importazione in background con avanzamento e annullamento, un flusso di lavoro con un errore compensato, la pagina di diagnostica e il pacchetto di deployment con il suo rollback.

Ogni tappa abbina una schermata funzionante a un percorso di errore, perché la revisione riguarda ciò che succede quando le cose vanno storte. Il revisore chiederà dove risiede lo stato, chi può chiamare che cosa, come si annulla un rilascio sbagliato e che cosa farebbe il team alle due di notte. La dichiarazione di prontezza per la produzione risponde per iscritto a queste domande, elenca ciò che non è ancora fatto e nomina il responsabile di ogni punto aperto.

Una difesa riesce quando ogni decisione è riconducibile a un motivo, ogni rischio ha una mitigazione con un nome e il team sa dire chiaramente che cosa cambierebbe dopo.



La regola di produzione avanzata

Non fermarti quando la schermata sembra corretta. Prima di considerare completo il modulo, spiega a chi appartiene lo stato, le implicazioni di sicurezza, i percorsi di errore e il comportamento in produzione.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Usare l'assistenza dell'IA per accelerare lo sviluppo Wisej.NET senza accettare codice non revisionato.

- ✓ Creare prompt e checklist di revisione basati sugli standard di progetto e sulla documentazione ufficiale.
- ✓ Preparare note di progetto pronte per MCP e per la documentazione, a uso dei futuri manutentori.
- ✓ Consegnare e difendere il capstone avanzato come una revisione dell'architettura di produzione.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.



Lab pratico

Completa il pacchetto del capstone. Aggiungi note sull'uso dell'IA, una libreria di prompt, link alla documentazione, una checklist di code review, lo script della demo finale, lo schema della presentazione di difesa dell'architettura e una dichiarazione di prontezza per la produzione.

Approccio suggerito

- ✓ Leggi il problema in produzione e individua il rischio architetturale prima di toccare la UI.
- ✓ Crea o aggiorna per primo l'incapsulamento in un servizio, comando, modello o componente coinvolto.
- ✓ Costruisci la schermata o il componente Wisej.NET seguendo gli standard di progetto stabiliti.
- ✓ Aggiungi una dimostrazione del percorso di errore, non solo del percorso felice.
- ✓ Annota che cosa è cambiato nel registro di apprendimento e aggiorna la checklist relativa.

Deliverable richiesti

- ✓ Libreria di prompt per l'IA
- ✓ Checklist di revisione del codice generato
- ✓ Indice della documentazione di progetto
- ✓ Script della demo del capstone
- ✓ Dichiarazione di prontezza per la produzione

Domande di revisione per lo studente

- ✓ Quale codice generato è stato accettato e perché?
- ✓ Ogni affermazione specifica della piattaforma è stata verificata sulla documentazione?
- ✓ Lo studente sa difendere le decisioni su deployment, sicurezza e stato?



La regola di produzione avanzata

Non fermarti quando la schermata sembra corretta. Prima di considerare completo il modulo, spiega a chi appartiene lo stato, le implicazioni di sicurezza, i percorsi di errore e il comportamento in produzione.