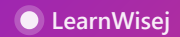


KI-Coding-Übung

Erstellen Sie das Beispielprojekt AccessOps dieses Moduls mit ChatGPT oder Claude aus einem fertigen Prompt, der das Modell auf die Spezifikation des Moduls verweist – und prüfen, starten und erweitern Sie dann, was Sie erhalten.



Dieses Beispiel tippen Sie nicht ab. Sie steuern es.

Kaum jemand schreibt Wisej.NET-Code heute noch in einer leeren Datei: Ein Assistent liefert den ersten Entwurf, und der Entwickler entscheidet, ob er stimmt. Das funktioniert aber nur, wenn Sie den Stoff beherrschen – und genau den hat Ihnen der Rest dieses Moduls vermittelt. Hier geben Sie dieses Wissen an ChatGPT oder Claude weiter, erhalten das Beispielprojekt des Moduls als lauffähige Visual-Studio-Projektmappe zurück und prüfen es so, wie Sie den Pull-Request eines Kollegen prüfen würden.

Worum Sie bitten

Ergänzen Sie AccessOps um Auditing und Secret-Verwaltung: einen AuditService, der Genehmigungen, Berechtigungsänderungen und Dokument-Downloads mit Benutzer, Uhrzeit, Adresse und Korrelations-ID festhält, eine Append-only-Audit-Tabelle, eine pro Befehl erzeugte Korrelations-ID, die in jede Logzeile geschrieben wird, Connection-Strings und das OIDC-Client-Secret, die in die Umgebungskonfiguration verlagert werden, mit einem dokumentierten Rotationsverfahren, und ein Log-Review, das sensible Felder schwärzt.

Das Modul verlangt diese Ergebnisse, und der Prompt unten fordert genau diese vom Assistenten an:

- ✓ AuditService und Append-only-Audit-Tabelle
- ✓ Korrelations-ID in jedem Befehl und jeder Logzeile
- ✓ In die Umgebungskonfiguration verlagerte Secrets
- ✓ Verfahren zur Rotation von Secrets
- ✓ Review der Log-Schwärzung

Warum es so aufgebaut ist

AccessOps authentifiziert inzwischen Personen, löst ihre Berechtigungen auf und schützt seine Dokumente – und nichts davon hilft an dem Morgen, an dem jemand fragt, wer Antrag 4471 genehmigt hat, von welcher Adresse aus und ob vorher jemand abgewiesen wurde. Die vorigen fünf Module haben entschieden, was erlaubt ist; dieses entscheidet, was sich im Nachhinein nachvollziehen lässt. Außerdem schließt es die Lücke, die aus einem kleinen Vorfall einen großen macht: ein Datenbankpasswort oder ein OIDC-Client-Secret in einer Datei, die vor zwei Jahren eingecheckt wurde.

Diese Begriffe muss der Assistent richtig verwenden. Geht die Antwort nachlässig mit ihnen um, hat er das Modul nicht verstanden:

Begriff	Was er hier bedeutet
Audit-Trail	Ein geschäftlicher Nachweis jedes sensiblen Befehls – wer gehandelt hat, woran, wann und von wo –, aufbewahrt für eine festgelegte Frist, weil sich jemand dafür wird verantworten müssen.
Append-only-Speicher	Eine Audit-Tabelle, in die das Anwendungskonto nur per INSERT schreiben darf. Wenn der Code, der die Historie schreibt, sie auch umschreiben kann, beweist die Historie nichts.
Manipulationsnachweis	Eine lückenlose Sequenz plus eine Hash-Kette, bei der der Hash jeder Zeile ihre eigenen Felder und den vorherigen Hash abdeckt, sodass eine bearbeitete Zeile die Kette bricht.
Korrelations-ID	Ein Bezeichner, der beim Start eines Befehls erzeugt und auf jeder Logzeile und jeder Audit-Zeile wiederholt wird, die dieser Befehl verursacht, sodass sich ein einzelner Klick später rekonstruieren lässt.
Logging-Scope	Ein umgebender Eigenschaftsblock – ILogger.BeginScope oder ein Serilog-Enricher –, der die Korrelations-ID und den Akteur an alles anhängt, was darin protokolliert wird.
Secret-Speicher	Der Ort, an dem ein Credential außerhalb des Builds lebt: lokal User Secrets, in einem Container Umgebungsvariablen, im Produktivbetrieb ein verwalteter Vault – oder eine Managed Identity, bei der überhaupt nichts gespeichert wird.
Schwärzung	Eine Regel dafür, was nie in einer Log-Senke landet – Passwörter, Tokens, Cookies, Connection-Strings und personenbezogene Daten –, durchgesetzt durch strukturiertes Logging, nicht durch Erinnerungsvermögen.

Selbst bauen – wählen Sie Ihren Assistenten

Beide Buttons öffnen einen neuen Tab, in dem **derselbe fertige Prompt** bereits eingetragen ist. Der Prompt verweist das Modell auf die Spezifikation dieses Moduls – eine einzelne Markdown-Datei mit der Lektion, der Lab-Aufgabe und den geforderten Ergebnissen –, sodass es mit dem echten Material arbeitet, statt zu raten, worum es in Modul 6 geht. Sie können [module-06.md](#) selbst öffnen, um genau zu sehen, was es lesen wird.



ChatGPT verwenden

Öffnet chatgpt.com mit dem Prompt dieses Moduls. Funktioniert mit GPT-5 oder Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.



Claude verwenden

Öffnet claude.ai mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie ihn stattdessen in einem leeren Ordner in ein Terminal einfügen.



Unverändert abgeschickt, erhalten Sie ein Projekt, kein Code-Fragment

Der Prompt verlangt am Ende die gesamte Projektmappe als einzelne `.zip`-Datei mit Anleitung zum Kompilieren und Starten. Laden Sie sie herunter, öffnen Sie die Projektmappe in Visual Studio, drücken Sie F5, und AccessOps startet auf `localhost`. Sie benötigen das .NET SDK und das NuGet-Paket `Wisej-4`; alles andere soll der Assistent laut Prompt selbst schreiben.

Ich arbeite gerade Modul 6 – „Auditing, Secrets und Konfiguration“ – des LearnWisej-Kurses „Sicherheits- und Authentifizierungsmuster“ durch und möchte das Beispielprojekt des Moduls gemeinsam mit dir erstellen. Bitte antworte auf Deutsch.

Lies zuerst die Modulspezifikation. Sie ist eine einzelne Markdown-Datei (auf Englisch) mit der vollständigen Lektion, der Lab-Aufgabe und den geforderten Ergebnissen:

<https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-06.md>

Erstelle dann ein vollständiges Wisej.NET-4-Projekt, das ich in Visual Studio öffnen kann:

1. Projekt „AccessOps“, C#, net8.0 oder neuer, NuGet-Paket Wisej-4 4.1.x, Kestrel-Host, der `app.UseWisej()` aufruft.
2. Setze um, was das Lab der Spezifikation verlangt: Ergänzen Sie AccessOps um Auditing und Secret-Verwaltung: einen `AuditService`, der Genehmigungen, Berechtigungsänderungen und Dokument-Downloads mit Benutzer, Uhrzeit, Adresse und Korrelations-ID festhält, eine `Append-only-Audit-Tabelle`, eine pro Befehl erzeugte Korrelations-ID, die in jede Logzeile geschrieben wird, `Connection-Strings` und das `OIDC-Client-Secret`, die in die Umgebungskonfiguration verlagert werden, mit einem dokumentierten Rotationsverfahren, und ein `Log-Review`, das sensible Felder schwärzt.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `AuditService` und `Append-only-Audit-Tabelle`
 - Korrelations-ID in jedem Befehl und jeder Logzeile
 - In die Umgebungskonfiguration verlagerte Secrets
 - Verfahren zur Rotation von Secrets
 - Review der Log-Schwärzung
4. Halte dich an die Spezifikation, wo immer sie eine Wisej.NET-Klasse, -Eigenschaft oder ein -Event nennt – verwende genau diese. Wo sie eine Wahl offenlässt, sag mir, welche API du gewählt hast und warum.
5. Der Server bleibt maßgeblich: Zustand und Validierung leben in .NET, der Browser stellt dar und meldet zurück. Verlagere nie eine Entscheidung nach JavaScript, nur weil es dort einfacher ist.
6. Mach das Ergebnis vorführbar: Ich muss es starten und das Verhalten sehen können, um das es in diesem Modul geht – auch, was passiert, wenn etwas schiefgeht.

Bevor du Code schreibst, fasse in eigenen Worten zusammen, was du bauen wirst und wo jedes Teil liegen wird, damit ich den Entwurf korrigieren kann, solange das noch wenig kostet.

Wenn der Code fertig ist, packe die gesamte Projektmappe in eine einzelne .zip-Datei zum Herunterladen und nenne mir die genauen Schritte, um sie zu kompilieren und auszuführen.

Beantworte zum Schluss diese Fragen aus der Spezifikation und nutze dabei deine eigene Implementierung als Beleg:

- Ihre `Audit-Tabelle` ist für die Anwendung nur beschreibbar per `Insert`, und jede Zeile ist per Hash verkettet. Nennen Sie etwas, das ein Angreifer mit dem eigenen Datenbank-Login der Anwendung dem Trail trotzdem antun kann, und erklären Sie genau, wie der Prüfdurchlauf es sichtbar machen würde.
- Eine Genehmigung wird auf dem Genehmigungsbildschirm gestartet und in einem mit `Application.StartTask` gestarteten Hintergrund-Task abgeschlossen, nachdem sich der Benutzer abgemeldet hat. Woher bekommt dieser Task seine Korrelations-ID und seinen Akteur, und was geht kaputt, wenn er sie stattdessen aus der Session liest?
- Ein Entwickler entfernt das `OIDC-Client-Secret` aus der versionierten Konfigurationsdatei und pusht die Änderung. Warum ist das Secret trotzdem noch kompromittiert, und was muss

geschehen, bevor der Vorfall tatsächlich abgeschlossen ist?

Frag mich nach allem, was unklar ist, bevor du anfängst.

Prüfen Sie dann, was zurückkommt

Code, der kompiliert, ist noch lange kein richtiger Code. Die spannenden Fragen sind Designfragen: Ist das wirklich das Wisej.NET-Control, das die Aufgabe verlangt? Folgt der Code dem Lebenszyklus, den das Framework vorgibt? Liegt der maßgebliche Zustand auf dem Server – oder wurde er im Browser abgelegt, weil das bequemer war? Lesen Sie das Ergebnis als Reviewer, nicht als Kunde.

Die Checkliste liefert Ihnen dieses Modul. Beantworten Sie jeden Punkt anhand des Codes, den Sie gerade erhalten haben:

- ✓ Ihre Audit-Tabelle ist für die Anwendung nur beschreibbar per Insert, und jede Zeile ist per Hash verkettet. Nennen Sie etwas, das ein Angreifer mit dem eigenen Datenbank-Login der Anwendung dem Trail trotzdem antun kann, und erklären Sie genau, wie der Prüfdurchlauf es sichtbar machen würde.
- ✓ Eine Genehmigung wird auf dem Genehmigungsbildschirm gestartet und in einem mit `Application.StartTask` gestarteten Hintergrund-Task abgeschlossen, nachdem sich der Benutzer abgemeldet hat. Woher bekommt dieser Task seine Korrelations-ID und seinen Akteur, und was geht kaputt, wenn er sie stattdessen aus der Session liest?
- ✓ Ein Entwickler entfernt das OIDC-Client-Secret aus der versionierten Konfigurationsdatei und pusht die Änderung. Warum ist das Secret trotzdem noch kompromittiert, und was muss geschehen, bevor der Vorfall tatsächlich abgeschlossen ist?

Alles, was Sie korrigieren mussten, sollten Sie festhalten. Eine kurze Notiz zu den Konventionen und Entscheidungen, auf die Sie sich festgelegt haben – eingefügt in die erste Nachricht Ihrer nächsten Sitzung –, ist mehr wert als jeder noch so clevere Prompt.



Für diesen Kurs gibt es noch kein Referenzbeispiel – Ihres ist das erste

Das [LearnWisej-Beispiel-Repository](#) enthält ausgearbeitete Builds für die meisten Kurse, für diesen aber noch nicht. Es gibt also nichts zum Vergleichen und keine Abkürzung: Die Spezifikation und Ihr eigenes Review sind das ganze Sicherheitsnetz. Behalten Sie, was Sie bauen – und wenn es gut wird, würden wir es gern sehen.

Drei Prompts für den nächsten Schritt

Richten Sie Ihren Assistenten auf das Projekt, das Sie gerade gebaut haben, und probieren Sie diese Prompts aus. Jeder ist eine Übung zum Prüfen und Erweitern, kein Neuschreiben.

- ✓ **Beweisen Sie, dass es wirklich funktioniert.** *"Geh dieses Projekt Ergebnis für Ergebnis anhand der Modulspezifikation unter <https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-06.md> durch und zeig mir den Beleg, dass jedes wirklich erledigt ist – den Codepfad, worauf ich klicken muss und was ich sehen sollte. Wo ein Ergebnis nur teilweise erfüllt ist, sag das klar, statt es schönzureden. Ergänze dann, was fehlt."*

- ✓ **Prüfen Sie es wie einen Pull-Request.** *"Führ mich durch diesen AccessOps-Code, als würdest du den Pull-Request eines Kollegen prüfen und hättest ihn nicht selbst geschrieben. Wo lebt der Zustand, was passiert bei einer einzelnen Benutzeraktion von Anfang bis Ende, was würdest du zur Überarbeitung zurückgeben, und welche Teile würdest du nicht guten Gewissens einem Kunden vorlegen? Sei konkret und zitiere die Zeilen, die du meinst – fasse mir nicht den Entwurf zusammen."*
- ✓ **Tauschen Sie ein Teil aus, ohne die Schnittstellen zu verschieben.** *"Wähle den Teil dieses AccessOps-Projekts, der am offensichtlichsten nur ein Platzhalter für etwas Echtes ist – eine Fake-Datenquelle, ein In-Memory-Speicher, ein vereinfachter Service – und ersetze ihn durch eine praxisnahe Implementierung. Der Vertrag auf beiden Seiten darf sich nicht ändern: dieselben Typen, dieselben Aufrufe, dasselbe Verhalten im Fehlerfall. Sag mir, was du durch die echte Implementierung behandeln musstest, was der Platzhalter nicht behandelt hat."*



Nehmen Sie es mit ins Lab

Das praktische Lab am Ende dieses Moduls verlangt dieselben Ergebnisse, die Sie gerade vom Assistenten haben erstellen lassen. Nutzen Sie, was Sie hier haben, als Gerüst für Ihre Antwort und als Beleg dafür – nicht als die Antwort selbst. Bewertet wird im Lab Ihre Argumentation, und dieser Teil muss von Ihnen stammen.