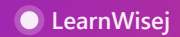


KI-Coding-Übung

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.



Instruieren, lesen, dann auf die Probe stellen.

Diese Übung hat drei Teile, und nur der erste dreht sich um den Prompt. Sie instruieren einen Assistenten mit der Modulspezifikation und erhalten eine AccessOps-Projektmappe zurück; Sie lesen, was er geschrieben hat, und entscheiden, ob es wirklich tut, was das Modul verlangt; dann treiben Sie es weiter, als das Modul verlangt hat. Im zweiten und dritten Teil steckt das eigentliche Lernen.

Worum Sie bitten

Ergänzen Sie AccessOps um Autorisierung: einen PermissionService, der beim Login Rollen und Claims in Berechtigungen auflöst, einen Authorize-Guard, den jede Methode von DocumentService und ApprovalService aufruft, eine Regel auf Datensatzebene, nach der Mitarbeitende nur ihre Abteilung und Manager nur ihre direkt Unterstellten sehen, eine UI, die Admin-Aktionen ausblendet und Approve-Buttons deaktiviert, die der Benutzer nicht verwenden darf, und einen Test, der belegt, dass der Service einen gefälschten Antrag von jemandem ohne Managerrolle abweist.

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

- ✓ PermissionService, der Berechtigungen in den SessionContext auflöst
- ✓ Autorisierungs-Guard in jeder Service-Methode
- ✓ Regeln auf Datensatzebene für Abteilung und Manager
- ✓ Berechtigungsabhängige Anpassung der UI
- ✓ Negativtest mit einem gefälschten Antrag

Warum es so aufgebaut ist

AccessOps weiß jetzt, wer angemeldet ist: Modul 2 hat eine verifizierte Identität im SessionContext hinterlassen. Zu wissen, wer jemand ist, beantwortet aber nicht, was diese Person mit einem vertraulichen Dokument oder einem sechsstelligen Genehmigungsantrag tun darf. Das typische Versagen in einem internen Portal ist selten ein kaputter Login – es ist der Sachbearbeiter der Lohnbuchhaltung, der die Verträge der Rechtsabteilung lesen kann, weil die Abfrage nie eingeschränkt wurde, oder die Teamleiterin, die ihre eigene Spesenabrechnung genehmigt, weil die einzige Prüfung ein deaktivierter Button war.

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
Rolle	Eine grobe, benannte Tätigkeitsfunktion wie Employee, Manager oder Auditor. Rollen sind bequem zu verwalten, aber zu grob, um direkt dagegen zu prüfen, denn ein Bildschirm muss über eine einzelne Aktion Bescheid wissen, nicht über eine ganze Stelle.
Claim	Eine Name-Wert-Aussage über die verifizierte Identität – Abteilung 14, Kostenstelle EU-North, Leitung von Team 7. Claims tragen die Daten, die Regeln auf Datensatzebene brauchen, und sie stammen aus dem Identitätsspeicher, nie aus dem Browser.
Berechtigung	Eine benannte Aktion, die die Anwendung ausführen kann, etwa ApproveRequest oder ViewDepartmentDocuments. Code prüft Berechtigungen, Administratoren vergeben Rollen, und die Zuordnung zwischen beiden ist Konfiguration.
Auflösung von Berechtigungen	Der einzelne Schritt beim Login, der Rollen und Claims zu einer flachen, unveränderlichen Berechtigungsmenge im SessionContext expandiert, sodass kein Request die Rollentabelle erneut liest und keine zwei Prüfungen zu unterschiedlichen Ergebnissen kommen.
Autorisierungs-Guard	Ein Aufruf wie Require(Permission.ApproveRequest) am Anfang einer Service-Methode, der eine Exception wirft, bevor irgendeine Arbeit geschieht. Er ist der einzige Ort, an dem tatsächlich durchgesetzt wird.
Regel auf Datensatzebene	Eine Einschränkung, welche Zeilen eine erlaubte Aktion berühren darf – die eigene Abteilung, die eigenen direkt unterstellten Mitarbeitenden –, angewendet innerhalb der Abfrage oder des Befehls statt erst, nachdem die Daten bereits geladen wurden.
UI-Anpassung	Eine Aktion ausblenden, deaktivieren oder erklären, basierend auf derselben Berechtigungsmenge, die der Server durchsetzt. Das ist eine Gefälligkeit gegenüber dem Benutzer und niemals eine Kontrolle.

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 3 geht. Sie können [module-03.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 3 – „Autorisierung: Rollen, Claims und Berechtigungen“ – 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-03.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 Autorisierung: einen `PermissionService`, der beim Login Rollen und Claims in Berechtigungen auflöst, einen `Authorize-Guard`, den jede Methode von `DocumentService` und `ApprovalService` aufruft, eine Regel auf Datenschutzebene, nach der Mitarbeitende nur ihre Abteilung und Manager nur ihre direkt Unterstellten sehen, eine UI, die Admin-Aktionen ausblendet und `Approve-Buttons` deaktiviert, die der Benutzer nicht verwenden darf, und einen Test, der belegt, dass der Service einen gefälschten Antrag von jemandem ohne Managerrolle abweist.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `PermissionService`, der Berechtigungen in den `SessionContext` auflöst
 - Autorisierungs-Guard in jeder Service-Methode
 - Regeln auf Datenschutzebene für Abteilung und Manager
 - Berechtigungsabhängige Anpassung der UI
 - Negativtest mit einem gefälschten Antrag
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:

- Eine Prüferin erhält Genehmigungsrechte, während ihre AccessOps-Session noch offen ist. Erklären Sie genau, wann die neue Berechtigung in Ihrem Design wirksam wird, warum Sie diesen Zeitpunkt gewählt haben und was Sie ändern müssten, damit sie sofort gilt.
- Ihre Dokumentabfrage filtert im Repository nach Abteilung. Beschreiben Sie, was ein Angreifer gewinnt, wenn dieser Filter stattdessen in die DataSource des Grids verlagert wird, obwohl das sichtbare Ergebnis auf dem Bildschirm identisch ist.
- Sowohl `ApprovalService` als auch sein Repository könnten prüfen, ob der Genehmigende der Vorgesetzte des Antragstellers ist. Begründen Sie, welcher der beiden der richtige Ort ist, und sagen Sie, was kaputtgeht, wenn die Prüfung im anderen oder in beiden liegt.

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:

- ✓ Eine Prüferin erhält Genehmigungsrechte, während ihre AccessOps-Session noch offen ist. Erklären Sie genau, wann die neue Berechtigung in Ihrem Design wirksam wird, warum Sie diesen Zeitpunkt gewählt haben und was Sie ändern müssten, damit sie sofort gilt.
- ✓ Ihre Dokumentabfrage filtert im Repository nach Abteilung. Beschreiben Sie, was ein Angreifer gewinnt, wenn dieser Filter stattdessen in die DataSource des Grids verlagert wird, obwohl das sichtbare Ergebnis auf dem Bildschirm identisch ist.
- ✓ Sowohl ApprovalService als auch sein Repository könnten prüfen, ob der Genehmigende der Vorgesetzte des Antragstellers ist. Begründen Sie, welcher der beiden der richtige Ort ist, und sagen Sie, was kaputtgeht, wenn die Prüfung im anderen oder in beiden liegt.

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.

- ✓ **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."*
- ✓ **Machen Sie es fit für einen schlechten Tag.** *"Untersuche dieses AccessOps-Projekt auf Fehlerszenarien, die es noch nicht behandelt – fehlerhafte oder fehlende Eingaben, eine*

Abhängigkeit, die langsam oder nicht erreichbar ist, Arbeit, die eintrifft, nachdem der Benutzer die Seite verlassen hat, und zwei Aktionen, die sich gegenseitig in die Quere kommen. Sag mir für jedes zuerst, ob es in Wisej.NET 4 ein echtes Risiko ist und warum, und schreib erst dann die Korrektur. Wo du entscheidest, dass nichts behandelt werden muss, sag das und begründe es, statt defensiven Code hinzuzufügen."

- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses AccessOps-Projekt um einen zweiten Fall, der demselben Muster folgt wie der erste – eine weitere Entität, Maske oder ein Control derselben Form, auf dieselbe Weise verdrahtet. Lagere wirklich Gemeinsames in eine Basisklasse oder einen Service aus, aber nur dort, wo es sich lohnt; zwei ähnliche Dinge brauchen nicht immer eine Abstraktion. Zeig mir den Diff und begründe jede Auslagerung."*



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.