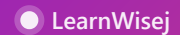


KI-Coding-Übung

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.



Den Code zu schreiben ist heute die leichtere Hälfte.

Funktionierenden Code für „Integration von OAuth und OpenID Connect“ bekommen Sie von einem modernen Assistenten mit einem guten Prompt. Ihn zu beurteilen, zu korrigieren und zu erweitern, erfordert alles, was dieses Modul behandelt hat. Genau auf dieser Aufteilung baut die Übung auf: Sie liefern Spezifikation und Review, der Assistent die Tipparbeit.

Worum Sie bitten

Ergänzen Sie AccessOps um eine externe Anmeldung: einen OpenID-Connect-Client, der den Authorization Code Flow mit PKCE gegen einen Test-Identity-Provider nutzt, einen Callback-Handler, der das ID-Token und seine Nonce validiert, ein Claim-Mapping, das den SessionContext befüllt und bei der ersten Anmeldung einen lokalen Mitarbeiterdatensatz anlegt, einen Rückfall auf das lokale Login für Service-Konten und eine Abmeldung, die sowohl die lokale als auch die Provider-Session beendet.

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

- ✓ OIDC-Client-Konfiguration mit PKCE
- ✓ Callback-Verarbeitung mit Token- und Nonce-Validierung
- ✓ Claim-Mapping und Provisionierung bei der ersten Anmeldung
- ✓ Rückfall auf lokales Login für Service-Konten
- ✓ Abmeldung, die beide Sessions abdeckt

Warum es so aufgebaut ist

AccessOps kann bereits ein eigenes Passwort prüfen, doch die Organisation, die es betreibt, will kein zweites Verzeichnis: Die Belegschaft meldet sich überall sonst beim Identity Provider des Unternehmens an, und Eintritte, Austritte und Multi-Faktor-Richtlinien werden an einer Stelle verwaltet. Den Login zu delegieren entfernt das Passwort vollständig aus AccessOps und ersetzt es durch eine andere Pflicht: Das Portal bekommt nun von einem Dritten über den Browser des Benutzers eine Identität übergeben und muss belegen, dass diese Identität echt, unverändert und für das Portal bestimmt ist, bevor auch nur ein einziger Abteilungsdatensatz angezeigt wird.

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
Authorization Code Flow	Der OpenID-Connect-Ablauf, bei dem der Browser immer nur einen kurzlebigen Einmal-Code transportiert, während Ihr Server die Tokens über einen direkten HTTPS-Aufruf abholt, den der Browser nie sieht.
PKCE	Ein zufälliger Code Verifier, der pro Anmeldung erzeugt, an den Authorize-Endpunkt nur als SHA-256-Hash gesendet und erst am Token-Endpunkt offengelegt wird, sodass ein abgefangener Code von niemand anderem eingelöst werden kann.
ID-Token	Ein signiertes JWT, das bestätigt, dass dieser Provider dieses Subjekt für diesen Client zu diesem Zeitpunkt authentifiziert hat. Es ist das einzige Token, das AccessOps heranzieht, um zu entscheiden, wer der Benutzer ist.
Access- und Refresh-Token	Das Access-Token ist ein Bearer-Credential für den Aufruf der API eines anderen; das Refresh-Token beschafft im Back-Channel Ersatz. Keines von beiden ist ein Identitätsnachweis für Ihre eigene App.
State und Nonce	Zwei Zufallswerte, die Sie pro Anmeldung erzeugen: State bindet den Callback an die Anfrage, die Sie tatsächlich gestartet haben, Nonce bindet das zurückgelieferte ID-Token an genau diese Anfrage, sodass kein älteres Token darin wiederverwendet werden kann.
Redirect-URI	Die exakte, vorab registrierte URL, zu der der Provider den Browser zurückschickt. In AccessOps ist das ein kleiner serverseitiger Endpunkt statt eines Wisej.NET-Fensters, weil er laufen muss, bevor irgendeine UI existiert.
Claim-Mapping und Provisionierung	Provider-Claims werden zu einem AccessOps-Mitarbeiter: Das Paar aus Issuer und Subject identifiziert das Konto, und bei der ersten Anmeldung wird eine lokale Zeile angelegt, die keine Berechtigungen trägt, bis jemand sie vergibt.

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 4 geht. Sie können [module-04.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 4 – „Integration von OAuth und OpenID Connect“ – 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-04.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 eine externe Anmeldung: einen OpenID-Connect-Client, der den Authorization Code Flow mit PKCE gegen einen Test-Identity-Provider nutzt, einen Callback-Handler, der das ID-Token und seine Nonce validiert, ein Claim-Mapping, das den SessionContext befüllt und bei der ersten Anmeldung einen lokalen Mitarbeiterdatensatz anlegt, einen Rückfall auf das lokale Login für Service-Konten und eine Abmeldung, die sowohl die lokale als auch die Provider-Session beendet.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - OIDC-Client-Konfiguration mit PKCE
 - Callback-Verarbeitung mit Token- und Nonce-Validierung
 - Claim-Mapping und Provisionierung bei der ersten Anmeldung
 - Rückfall auf lokales Login für Service-Konten
 - Abmeldung, die beide Sessions abdeckt
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:

- Ein Angreifer liest den Authorization Code aus dem Browserverlauf eines gemeinsam genutzten Rechners aus. Was hindert ihn daran, ihn gegen Tokens einzutauschen, und was hätte ihn ohne PKCE nicht aufgehalten?
- Ihr Provider stellt Gruppen-Claims aus, die genau wie die AccessOps-Berechtigungsnamen aussehen. Warum löst der PermissionService die Berechtigungen trotzdem selbst auf, statt diese Claims direkt in den SessionContext zu kopieren?
- Ein Benutzer klickt auf „Abmelden“, schließt nichts, klickt sofort wieder auf „Anmelden“ – und landet wieder in AccessOps, ohne etwas eingeben zu müssen. Welche Session hat überlebt, und welcher Redirect fehlte?

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:

- ✓ Ein Angreifer liest den Authorization Code aus dem Browserverlauf eines gemeinsam genutzten Rechners aus. Was hindert ihn daran, ihn gegen Tokens einzutauschen, und was hätte ihn ohne PKCE nicht aufgehalten?
- ✓ Ihr Provider stellt Gruppen-Claims aus, die genau wie die AccessOps-Berechtigungsnamen aussehen. Warum löst der PermissionService die Berechtigungen trotzdem selbst auf, statt diese Claims direkt in den SessionContext zu kopieren?
- ✓ Ein Benutzer klickt auf „Abmelden“, schließt nichts, klickt sofort wieder auf „Anmelden“ – und landet wieder in AccessOps, ohne etwas eingeben zu müssen. Welche Session hat überlebt, und welcher Redirect fehlte?

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.

- ✓ **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."*
- ✓ **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-04.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."*



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.