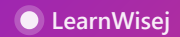


KI-Coding-Übung

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.



Der Assistent liefert die schnellen Hände. Der Architekt bleiben Sie.

Ein Assistent erstellt das Beispielprojekt dieses Moduls in wenigen Minuten. Was er nicht kann, ist zu entscheiden, ob das Ergebnis der richtige Entwurf für Ihre Anwendung ist – diese Entscheidung bleibt bei Ihnen. Das Urteilsvermögen bringen Sie mit: Instruieren Sie ihn sorgfältig und prüfen Sie dann, was zurückkommt, anhand dessen, was Ihnen Modul 7 gerade vermittelt hat.

Worum Sie bitten

Schließen Sie das AccessOps-Abschlussprojekt ab: HTTPS mit HSTS erzwingen und Security-Header setzen, Cookies als Secure und HttpOnly mit einer strikten SameSite-Richtlinie markieren, Session-Timeout und Request-Größenlimits festlegen, Rate Limiting für Login und Freigabelinks ergänzen, ein Audit der Abhängigkeiten durchführen und die Ergebnisse festhalten, dann die vollständige Checkliste dieses Kurses gegen die App durchgehen und jeden Befund beheben.

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

- ✓ Konfiguration von HTTPS, HSTS und Security-Headern
- ✓ Einstellungen zur Härtung von Cookies und Session
- ✓ Rate Limiting für Login und Freigabelinks
- ✓ Bericht zum Audit der Abhängigkeiten
- ✓ Ausgefüllte Checkliste des Sicherheits-Reviews mit Korrekturen

Warum es so aufgebaut ist

AccessOps prüft Passwörter inzwischen gegen einen gesalzenen Hash, löst Berechtigungen in einem Service auf, streamt Dokumente nur über DocumentService und protokolliert jede Genehmigung im Audit-Tail. Nichts davon hilft, wenn der erste Request im Klartext unterwegs ist, wenn ein Skript das Session-Cookie lesen kann oder wenn der Build ein Paket mit einer veröffentlichten Sicherheitsmeldung ausliefert. Die letzte Meile gehört dem Host: das Zertifikat, die Response-Header, die Cookie-Flags, die Request-Limits und die Version von allem im Abhängigkeitsgraphen.

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
HSTS	Ein Strict-Transport-Security -Header, der den Browser anweist, für diesen Host eine festgelegte Zeit lang kein einfaches HTTP zu verwenden, sodass ein späterer Besuch nie beim Redirect abgefangen wird.
Content-Security-Policy	Ein Header, der die Origins auflistet, von denen eine Seite Skripte, Styles und Frames laden darf; er macht aus einem eingeschleusten Skript einen blockierten Request statt Code, der in einer aktiven Session läuft.
frame-ancestors	Die CSP-Direktive (X-Frame-Options ist das ältere Gegenstück), die verhindert, dass eine andere Website AccessOps in einen Frame einbettet und einen Manager dazu bringt, auf Approve zu klicken.
Cookie-Flags	Secure hält das Session-Cookie von einfachem HTTP fern, HttpOnly hält es von JavaScript fern, und SameSite verhindert, dass es bei einem Request mitgeschickt wird, den eine andere Website gestartet hat.
Request-Limits	Obergrenzen für Body- und Header-Größe auf Host-Ebene, die einen übergroßen Upload abweisen, bevor auch nur eine Zeile AccessOps-Code läuft.
Token Bucket	Ein Zähler, der einen Versuch nur zulässt, solange er ein Token enthält, und sich mit fester Rate wieder auffüllt, sodass der Datenverkehr kurz stoßweise ansteigen darf und dann auf Schrittempo gebremst wird.
Audit der Abhängigkeiten	Eine wiederholbare Prüfung jedes direkten und transitiven Pakets sowie der Runtime gegen veröffentlichte Sicherheitsmeldungen – dokumentiert mit Datum und Entscheidung.

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 7 geht. Sie können [module-07.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 7 – „Härtungs-Checkliste und Abschlussprojekt“ – 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-07.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: Schließen Sie das AccessOps-Abschlussprojekt ab: HTTPS mit HSTS erzwingen und Security-Header setzen, Cookies als Secure und HttpOnly mit einer strikten SameSite-Richtlinie markieren, Session-Timeout und Request-Größenlimits festlegen, Rate Limiting für Login und Freigabelinks ergänzen, ein Audit der Abhängigkeiten durchführen und die Ergebnisse festhalten, dann die vollständige Checkliste dieses Kurses gegen die App durchgehen und jeden Befund beheben.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Konfiguration von HTTPS, HSTS und Security-Headern
 - Einstellungen zur Härtung von Cookies und Session
 - Rate Limiting für Login und Freigabelinks
 - Bericht zum Audit der Abhängigkeiten
 - Ausgefüllte Checkliste des Sicherheits-Reviews mit Korrekturen
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:

- Middleware kann das Rate Limiting für den Handler der Freigabelinks übernehmen, aber nicht für den Sign-in-Button. Was unterscheidet die beiden Requests in einer Wisej.NET-App, und wo muss das Login-Limit stattdessen leben?
- Ihre Content-Security-Policy wird durchgesetzt, und das Grid wird ohne Styles dargestellt. Welche Direktive ist die wahrscheinliche Ursache, und wie hätte Ihnen der Report-only-Modus das mitgeteilt, bevor es das Deployment tat?
- Das Audit der Abhängigkeiten listet ein verwundbares Paket auf, das in AccessOps niemand direkt referenziert. Wie ist es in den Build gelangt, und welche zwei ehrlichen Wege gibt es, den Befund zu schließen?

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:

- ✓ Middleware kann das Rate Limiting für den Handler der Freigabelinks übernehmen, aber nicht für den Sign-in-Button. Was unterscheidet die beiden Requests in einer Wisej.NET-App, und wo muss das Login-Limit stattdessen leben?
- ✓ Ihre Content-Security-Policy wird durchgesetzt, und das Grid wird ohne Styles dargestellt. Welche Direktive ist die wahrscheinliche Ursache, und wie hätte Ihnen der Report-only-Modus das mitgeteilt, bevor es das Deployment tat?
- ✓ Das Audit der Abhängigkeiten listet ein verwundbares Paket auf, das in AccessOps niemand direkt referenziert. Wie ist es in den Build gelangt, und welche zwei ehrlichen Wege gibt es, den Befund zu schließen?

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.

- ✓ **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."

- ✓ **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."*



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.