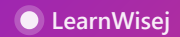


KI-Coding-Übung

Erstellen Sie das Beispielprojekt FieldOps 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 FieldOps-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

Paketieren Sie FieldOps für Windows: eine WebView2-Shell mit nativem Fenster, eigenem Titel und Tray-Icon, eine Bridge, über die die App einen nativen Ordnerdialog zum Export von Auftragsberichten öffnen und bei Eingang eines neuen Auftrags eine native Benachrichtigung anzeigen kann, eine Konfiguration, die die Shell auf den Remote-Server mit lokalem Fallback richtet, und ein Installer mit Update-Prüfung beim Start.

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

- ✓ WebView2-Desktop-Shell mit Fenster und Tray
- ✓ Native Bridge für Ordnerauswahl und Benachrichtigungen
- ✓ Konfiguration für Remote- und lokalen Server
- ✓ Installer-Paket
- ✓ Update-Prüfung beim Start

Warum es so aufgebaut ist

Die Techniker von FieldOps verbringen ihre gesamte Schicht in der App: ein Laptop im Transporter, eine Dockingstation im Depot. Der Browser-Build aus den ersten beiden Modulen erledigt die Arbeit, aber er hat kein Icon im Startmenü, keine Präsenz im Infobereich, wenn ein Auftrag zugewiesen wird, keine Möglichkeit, einen exportierten Bericht in einem vom Techniker gewählten Ordner abzulegen, und nichts, was die IT über ihren üblichen Softwarekanal verteilen könnte. Dieses Modul verpackt dieselbe Anwendung unverändert in eine Windows-Shell, die genau diese vier Dinge liefert.

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
WebView2-Control	Die Chromium-Engine, die Microsoft als weiterverteilbare Evergreen Runtime ausliefert, eingebettet in ein WinForms- oder WPF-Fenster, sodass Ihre Seite genau wie in Edge gerendert wird.
Desktop-Shell	Eine kleine native ausführbare Datei, die das Fenster, das Tray-Icon und die Betriebssystemaufrufe besitzt und ein WebView2-Control auf Ihre Wisej.NET-Anwendung richtet.
Fensterrahmen	Titel, Icon, Mindestgröße, gespeicherte Position und Einzelinstanz-Verhalten – all das, was die App installiert statt eingebettet wirken lässt.
Nachrichten-Bridge	Der bidirektionale Kanal zwischen Seite und Shell: PostWebMessageAsJson nach unten, WebMessageReceived nach oben, oder ein mit AddHostObjectToScript veröffentlichtes Host-Objekt.
JavaScript-Interop	Wie serverseitiger C#-Code diesen Kanal erreicht: Application.Eval und Application.Call führen Skript im Browserkontext der Session aus, und die Antwort kommt als Client-Event zurück.
Server-Topologie	Ob die Shell zu einem entfernten Unternehmensserver navigiert oder einen selbst gehosteten Wisej.NET-Prozess auf Loopback startet – beim Start entschieden und protokolliert, niemals geraten.
Update-Kanal	Ein Versions-Endpunkt, den die Shell beim Start abfragt, plus der Installer, den sie herunterlädt, wenn die Antwort neuer ist als die laufende Version.

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 FieldOps 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 – „Desktop-Paketierung mit einer WebView-Shell“ – des LearnWisej-Kurses „Hybride und native Paketierung“ 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/hybrid-native-packaging/specs/module-03.md>

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

1. Projekt „FieldOps“, 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: Paketieren Sie FieldOps für Windows: eine WebView2-Shell mit nativem Fenster, eigenem Titel und Tray-Icon, eine Bridge, über die die App einen nativen Ordnerdialog zum Export von Auftragsberichten öffnen und bei Eingang eines neuen Auftrags eine native Benachrichtigung anzeigen kann, eine Konfiguration, die die Shell auf den Remote-Server mit lokalem Fallback richtet, und ein Installer mit Update-Prüfung beim Start.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - WebView2-Desktop-Shell mit Fenster und Tray
 - Native Bridge für Ordnerauswahl und Benachrichtigungen
 - Konfiguration für Remote- und lokalen Server
 - Installer-Paket
 - Update-Prüfung beim Start
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 Techniker startet zwei Exporte innerhalb einer Sekunde, und die Shell beantwortet beide Ordnerdialoge. Was in Ihrem Bridge-Design verhindert, dass der zweite Pfad in den ersten Bericht geschrieben wird, und was würde ohne diese Maßnahme schiefgehen?
- Ihr Export-Button öffnet jetzt einen nativen Ordnerdialog. Beschreiben Sie genau, was derselbe Button tut, wenn die Seite in Edge statt in der Shell geöffnet wird, und wo in Ihrem Code dieser Unterschied entschieden wird.
- Der Versions-Endpunkt ist nicht erreichbar, weil der Transporter keinen Empfang hat. Begründen Sie, was die Shell bei diesem Start tun sollte, und erklären Sie, was ein Techniker bei jeder der verworfenen Alternativen verliert.

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 Techniker startet zwei Exporte innerhalb einer Sekunde, und die Shell beantwortet beide Ordnerdialoge. Was in Ihrem Bridge-Design verhindert, dass der zweite Pfad in den ersten Bericht geschrieben wird, und was würde ohne diese Maßnahme schiefgehen?
- ✓ Ihr Export-Button öffnet jetzt einen nativen Ordnerdialog. Beschreiben Sie genau, was derselbe Button tut, wenn die Seite in Edge statt in der Shell geöffnet wird, und wo in Ihrem Code dieser Unterschied entschieden wird.
- ✓ Der Versions-Endpunkt ist nicht erreichbar, weil der Transporter keinen Empfang hat. Begründen Sie, was die Shell bei diesem Start tun sollte, und erklären Sie, was ein Techniker bei jeder der verworfenen Alternativen verliert.

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.

- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses FieldOps-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/hybrid-native-packaging/specs/module-03.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 FieldOps-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."*



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.