

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.



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

Erstellen Sie die FieldOps-Projektmappe mit einer Liste der Arbeitsaufträge, einer Detailseite für einen Arbeitsauftrag und einem Ablauf zur Statusaktualisierung, die in einem einfachen Browser funktionieren. Schreiben Sie das Entscheidungsprotokoll zur Bereitstellung: Vergleichen Sie Browser, PWA, Desktop- und mobile Shell anhand der Anforderungen der Techniker, wählen Sie die Zielplattformen und ihre Reihenfolge, und definieren Sie die Capability-Checks, mit denen die App Gerätefunktionen aktiviert.

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

- ✓ FieldOps-Projektmappe mit Liste, Detailseite und Statusaktualisierung
- ✓ Vergleichstabelle der Bereitstellungsmodelle
- ✓ Entscheidungsprotokoll zur Bereitstellung mit Reihenfolge der Zielplattformen
- ✓ Entwurf des Capability-Checks
- ✓ Architekturdiagramm mit einem Server und mehreren Shells

Warum es so aufgebaut ist

FieldOps ist eine Außendienstanwendung: Techniker erhalten Arbeitsaufträge, nehmen Fotos und Unterschriften auf, navigieren zu Einsatzorten und arbeiten mit lückenhafter Verbindung. Als gewöhnliche Wisej.NET-Anwendung läuft sie bereits in jedem Browser – das genügt den Disponenten im Büro, aber nicht dem Techniker, der mit einem Balken Empfang und dem Handy im Handschuh im Technikraum steht. Die Frage dieses Moduls lautet nicht „Können wir das bauen?“, sondern „Wie erreicht die App die Menschen, die sie nutzen?“ – und die Antwort darauf ist eine Architekturentscheidung, kein nachträglicher Packaging-Schritt.

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
Bereitstellungsmodell	Der Weg, auf dem die Anwendung zum Benutzer gelangt: ein Browser-Tab, eine installierte PWA, ein Desktop-Fenster oder eine über den Store verteilte mobile App. Es verändert Verteilung, Updates und Gerätezugriff – niemals den C#-Code, der den Bildschirm aufbaut.
Browser-Engine des Hosts	Jedes Modell rendert dieselbe servergesteuerte Oberfläche in einer Browser-Engine: im eigenen Browser des Benutzers oder in einer eingebetteten Engine – WebView2 unter Windows, WebKit unter macOS und iOS, die System-WebView unter Android.
Wisej.NET Hybrid	Die auf .NET MAUI basierende Shell des Frameworks, die Ihre bestehende Wisej.NET-Anwendung unter Android, iOS und auf dem Desktop hostet und eine Bridge zu Gerätefunktionen wie Kamera, Geolocation, Dateien und Benachrichtigungen ergänzt.
Geräte-Bridge	Der bidirektionale Kanal zwischen Ihrem serverseitigen C#-Code und der nativen Shell: Der Server fordert eine Geräteaktion an, die Shell führt sie aus, und das Ergebnis kommt als Event in Ihren Code zurück, nicht als Rückgabewert.
Capability-Check	Eine Laufzeitprüfung dessen, was die aktuelle Shell tatsächlich kann. Damit wird eine Funktion angezeigt, ausgeblendet oder ersetzt, sodass sich ein Bildschirm in jedem Bereitstellungsmodell korrekt verhält.
Cache der Anwendungs-Shell	Das statische HTML, CSS, die Skripte und Icons, die ein PWA-Service-Worker lokal vorhalten kann. Das ist der Teil, der auch ohne Netzwerk noch lädt – der interaktive, serverseitig gerenderte Teil der App gehört nicht dazu.
Update-Asymmetrie	Der Server kann jederzeit neu bereitgestellt werden, und jede Session erhält beim Neuladen den neuen Build – ein Desktop-Installer oder ein Store-Paket dagegen braucht zuerst ein Release und eine Installation.

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 1 geht. Sie können [module-01.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 1 – „Hosting-Modelle jenseits des Browsers“ – 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-01.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: Erstellen Sie die FieldOps-Projektmappe mit einer Liste der Arbeitsaufträge, einer Detailseite für einen Arbeitsauftrag und einem Ablauf zur Statusaktualisierung, die in einem einfachen Browser funktionieren. Schreiben Sie das Entscheidungsprotokoll zur Bereitstellung: Vergleichen Sie Browser, PWA, Desktop- und mobile Shell anhand der Anforderungen der Techniker, wählen Sie die Zielplattformen und ihre Reihenfolge, und definieren Sie die Capability-Checks, mit denen die App Gerätefunktionen aktiviert.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - FieldOps-Projektmappe mit Liste, Detailseite und Statusaktualisierung
 - Vergleichstabelle der Bereitstellungsmodelle
 - Entscheidungsprotokoll zur Bereitstellung mit Reihenfolge der Zielplattformen
 - Entwurf des Capability-Checks
 - Architekturdiagramm mit einem Server und mehreren Shells
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 verliert mitten in der Aktualisierung eines Arbeitsauftrags den Empfang. Beschreiben Sie genau, welche Teile von FieldOps noch aus einer gecachten Shell angezeigt werden können, welche nicht, und was Ihr Bildschirm in dem Moment tun sollte, in dem die Verbindung abbricht.
- Ihr Entscheidungsprotokoll stellt die Android-Shell vor die PWA. Was müsste für die Geräte und das Netz der Techniker gelten, damit diese Reihenfolge falsch wäre?
- Die Detailseite braucht ein Foto. Erklären Sie, warum ein Capability-Check mit einem Upload-Fallback ein besseres Design ist als ein Build-Schalter, der eine reine Mobilversion des Bildschirms erzeugt.

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 verliert mitten in der Aktualisierung eines Arbeitsauftrags den Empfang. Beschreiben Sie genau, welche Teile von FieldOps noch aus einer gecachten Shell angezeigt werden können, welche nicht, und was Ihr Bildschirm in dem Moment tun sollte, in dem die Verbindung abbricht.
- ✓ Ihr Entscheidungsprotokoll stellt die Android-Shell vor die PWA. Was müsste für die Geräte und das Netz der Techniker gelten, damit diese Reihenfolge falsch wäre?
- ✓ Die Detailseite braucht ein Foto. Erklären Sie, warum ein Capability-Check mit einem Upload-Fallback ein besseres Design ist als ein Build-Schalter, der eine reine Mobilversion des Bildschirms erzeugt.

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 FieldOps-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 FieldOps-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 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."*



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.