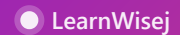


KI-Coding-Übung

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.



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 2 gerade vermittelt hat.

Worum Sie bitten

Machen Sie FieldOps zu einer PWA: Fügen Sie das Manifest mit Icons, Theme-Farbe und Standalone-Anzeige hinzu, registrieren Sie einen Service Worker, der die Shell und statische Assets cacht, zeigen Sie nach der ersten erfolgreichen Session einen Installationsdialog an, blenden Sie eine Offline-Seite mit der Zusammenfassung der zuletzt synchronisierten Arbeitsaufträge ein, wenn der Server nicht erreichbar ist, und implementieren Sie den Update-Ablauf, der neu lädt, sobald eine neue Version verfügbar ist.

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

- ✓ Web-App-Manifest mit Icons und Farben
- ✓ Service Worker, der die Shell cacht
- ✓ Installationsdialog nach der ersten Session
- ✓ Offline-Seite mit der zuletzt synchronisierten Zusammenfassung
- ✓ Update-Ablauf und Versionsprüfung

Warum es so aufgebaut ist

Die Techniker von FieldOps öffnen keinen Browser und tippen keine URL ein. Sie tippen auf ein Icon auf einem Smartphone, das den ganzen Vormittag im Transporter lag, in einem Technikraum, in dem das Signal mitten im Einsatz auf null fällt. Eine Progressive Web App gibt ihnen dieses Icon, ein Fenster ohne Browser-Bedienelemente und eine Seite, die noch etwas Nützliches sagt, wenn das Netz verschwindet – aber FieldOps ist eine Wisej.NET-App, ihre Benutzeroberfläche lebt also auf dem Server, und wer so tut, als wäre es anders, macht aus der PWA ein Support-Ticket.

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
Web-App-Manifest	Eine kleine JSON-Datei, auf die die Startseite verlinkt und die dem Browser Namen, Icons, Farben, Anzeigemodus und Start-URL der App mitteilt – ohne sie hat der Browser nichts zu installieren.
Anzeigemodus	Das display -Feld des Manifests. standalone startet FieldOps in einem eigenen Fenster ohne Adressleiste – erst dadurch fühlt sich eine installierte App nicht mehr wie ein Lesezeichen an.
Maskierbare Icons	Icons, die mit purpose: "maskable" und ausreichendem Innenabstand deklariert sind, damit Android sie in seine eigene Form zuschneiden kann, ohne die Grafik abzuschneiden.
Service Worker	Ein Skript, das der Browser neben der Seite ausführt, von der eigenen Origin der App über HTTPS ausgeliefert, und das Anfragen der Seite aus einem Cache beantworten kann, wenn die Verbindung langsam oder weg ist.
Anwendungs-Shell	Die statischen Dateien, die eine Session braucht, bevor der Server überhaupt etwas sagen kann: die Startseite, die Client-Skripte, das Theme und die Icons. Nur sie darf der Worker cachen.
Cache-First vs. Network-Only	Die Routing-Regel im Fetch-Handler des Workers: Statische Shell-Dateien kommen zuerst aus dem Cache, während alles, was die Wisej.NET-Session mit dem Server austauscht, direkt ins Netzwerk gehen muss – nie in eine gecachte Kopie.
Update-Lebenszyklus	Ein neuer Worker installiert sich, während der alte die Seite noch kontrolliert, und wartet dann. skipWaiting, clients.claim und ein kontrolliertes Neuladen machen aus diesem wartenden Worker die Version, die der Techniker tatsächlich ausführt.

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 2 geht. Sie können [module-02.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 2 – „Einrichtung einer Progressive Web App“ – 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-02.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: Machen Sie FieldOps zu einer PWA: Fügen Sie das Manifest mit Icons, Theme-Farbe und Standalone-Anzeige hinzu, registrieren Sie einen Service Worker, der die Shell und statische Assets cacht, zeigen Sie nach der ersten erfolgreichen Session einen Installationsdialog an, blenden Sie eine Offline-Seite mit der Zusammenfassung der zuletzt synchronisierten Arbeitsaufträge ein, wenn der Server nicht erreichbar ist, und implementieren Sie den Update-Ablauf, der neu lädt, sobald eine neue Version verfügbar ist.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Web-App-Manifest mit Icons und Farben
 - Service Worker, der die Shell cacht
 - Installationsdialog nach der ersten Session
 - Offline-Seite mit der zuletzt synchronisierten Zusammenfassung
 - Update-Ablauf und Versionsprüfung
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 Reviewer fragt, warum der Service Worker nicht einfach jede Antwort cacht, damit FieldOps auch unter der Erde weiter funktioniert. Erklären Sie, was eine wiedergegebene Antwort einer Wisej.NET-Session tatsächlich mit dem Bildschirm des Technikers anrichten würde und wo die ehrliche Grenze verläuft.
- Ihr Installationsangebot funktioniert an Ihrem Schreibtisch, erscheint aber nie auf dem Smartphone des Technikers. Nennen Sie drei unterschiedliche Bedingungen, die dafür verantwortlich sein könnten, und sagen Sie, wie Sie sie am Gerät voneinander unterscheiden würden.
- Ein Techniker hat die App installiert und geöffnet, während Sie einen neuen Build deployen. Beschreiben Sie jeden Schritt zwischen Ihrem Deployment und dem Moment, in dem auf

diesem Smartphone die neue Shell läuft, und benennen Sie den Schritt, der stillschweigend scheitern würde, wenn sich der Cache-Name nie ändert.

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 Reviewer fragt, warum der Service Worker nicht einfach jede Antwort cacht, damit FieldOps auch unter der Erde weiter funktioniert. Erklären Sie, was eine wiedergegebene Antwort einer Wisej.NET-Session tatsächlich mit dem Bildschirm des Technikers anrichten würde und wo die ehrliche Grenze verläuft.
- ✓ Ihr Installationsangebot funktioniert an Ihrem Schreibtisch, erscheint aber nie auf dem Smartphone des Technikers. Nennen Sie drei unterschiedliche Bedingungen, die dafür verantwortlich sein könnten, und sagen Sie, wie Sie sie am Gerät voneinander unterscheiden würden.
- ✓ Ein Techniker hat die App installiert und geöffnet, während Sie einen neuen Build deployen. Beschreiben Sie jeden Schritt zwischen Ihrem Deployment und dem Moment, in dem auf diesem Smartphone die neue Shell läuft, und benennen Sie den Schritt, der stillschweigend scheitern würde, wenn sich der Cache-Name nie ändert.

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 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."*
- ✓ **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-02.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.