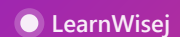


KI-Coding-Übung

Erstellen Sie das Beispielprojekt SupportDesk 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, auch im Vergleich mit unserem eigenen Referenz-Build auf GitHub.



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

Erstellen Sie die Entitäten Customer, Agent, Category, Ticket und TicketComment für die Support Desk Data Console, konfigurieren Sie Beziehungen und Löscherhalten mit der Fluent API, konfigurieren Sie RowVersion mit IsRowVersion, fügen Sie Indizes für Status, DueDate, CustomerId und UpdatedAt hinzu, ergänzen Sie eine Design-Time-Factory, erstellen Sie die Migration InitialCreate und befüllen Sie eine lokale SQLite- oder SQL-Server-LocalDB-Datenbank mit fünf Kunden, drei Agents, Kategorien und mindestens fünfzig Tickets.

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

- ✓ Entitäten Customer, Agent, Category, Ticket und TicketComment mit Beziehungen und Löscherhalten
- ✓ RowVersion mit IsRowVersion konfiguriert
- ✓ Indizes für Status, DueDate, CustomerId und UpdatedAt
- ✓ Design-Time-Factory und die Migration InitialCreate
- ✓ Seed-Daten: fünf Kunden, drei Agents, Kategorien und mindestens fünfzig Tickets

Warum es so aufgebaut ist

Die Support Desk Data Console zeigt Tickets in einem Grid mit Paging, bearbeitet sie in einem modalen Formular und lässt mehrere Agents gleichzeitig dieselbe Queue abarbeiten – und jede Oberfläche ist nur so gut wie das Modell darunter. Ein Filter ohne passenden Index wird bei fünfzigtausend Tickets zum Tabellenscan, zwei Agents, die dasselbe Ticket speichern, verlieren stillschweigend die Arbeit eines der beiden, wenn die Zeile kein Concurrency-Token trägt, und ein von Hand geflicktes Schema bricht das nächste Deployment. Das Modell ist der Vertrag zwischen UI und Datenbank; deshalb baut dieses Modul es explizit auf und versioniert es als Quellcode.

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
bindungsfreundliche Entität	Eine Entität mit vorhersehbarem Schlüssel, stabilen Anzeigefeldern, expliziten Navigationseigenschaften und kleinen Lookup-Tabellen, sodass ein Grid, eine ComboBox und ein Editor alle an sie binden können.
View-Modell TicketListItem	Eine flache Projektion nur der Spalten, die das Grid zeigt, sodass das Grid nie den ganzen Ticket-Graphen erhält und unabhängig von der Unit of Work zum Bearbeiten bleibt.
Konfiguration per Fluent API	Mapping-Regeln in OnModelCreating (Pflicht-Strings, Längen, Indizes, Löscherhalten, Genauigkeit, Standardwerte), sodass Persistenzregeln direkt beim DbContext liegen.
RowVersion als Concurrency-Token	Ein von der Datenbank erzeugtes Byte-Array, gekennzeichnet mit IsRowVersion; EF Core nimmt seinen Originalwert in die WHERE-Klausel des UPDATE auf, sodass ein veraltetes Speichern null Zeilen ändert und DbUpdateConcurrencyException auslöst.
Löscherhalten	Die OnDelete -Regel einer Beziehung (Restrict, SetNull, Cascade), die entscheidet, ob das Löschen eines Kunden abgelehnt wird, Tickets ihre Zuweisung verlieren oder auch Kommentare entfernt werden.
Migration	Eine C#-Klasse plus Modell-Snapshot, die eine geprüfte Schemaänderung beschreibt, in die Versionskontrolle committet und der Reihe nach angewendet wird, um jede Datenbank auf den Stand des Modells zu bringen.
Design-Time-Factory	Eine IDesignTimeDbContextFactory, die den dotnet ef -Tools sagt, wie sie den Context erzeugen, ohne den Startup der Wisej.NET-Anwendung auszuführen.

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 SupportDesk 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 – „Modellierung, DbContext-Konfiguration und Migrationen“ – des LearnWisej-Kurses „Data Binding mit EF Core“ 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/ef-core-binding/specs/module-02.md>

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

1. Projekt „SupportDesk“, 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 Entitäten Customer, Agent, Category, Ticket und TicketComment für die Support Desk Data Console, konfigurieren Sie Beziehungen und Löscherhalten mit der Fluent API, konfigurieren Sie RowVersion mit `IsRowVersion`, fügen Sie Indizes für Status, DueDate, CustomerId und UpdatedAt hinzu, ergänzen Sie eine Design-Time-Factory, erstellen Sie die Migration InitialCreate und befüllen Sie eine lokale SQLite- oder SQL-Server-LocalDB-Datenbank mit fünf Kunden, drei Agents, Kategorien und mindestens fünfzig Tickets.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Entitäten Customer, Agent, Category, Ticket und TicketComment mit Beziehungen und Löscherhalten
 - RowVersion mit `IsRowVersion` konfiguriert
 - Indizes für Status, DueDate, CustomerId und UpdatedAt
 - Design-Time-Factory und die Migration InitialCreate
 - Seed-Daten: fünf Kunden, drei Agents, Kategorien und mindestens fünfzig Tickets
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:

- Sie löschen einen Kunden, der noch offene Tickets hat. Was passiert mit dem von Ihnen konfigurierten Löscherhalten, und was wäre mit Cascade passiert?
- Zwei Agents laden das Ticket SD-1042 und klicken im Abstand von einer Sekunde auf Save. Welches Speichern löst eine Exception aus, was steht in der WHERE-Klausel der UPDATE-Anweisung, und was sieht der Benutzer?
- Warum liefert `IsRequired()` auf Title dem Benutzer keine Meldung „Enter a title“, und woher kommt diese Meldung später im Kurs?

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:

- ✓ Sie löschen einen Kunden, der noch offene Tickets hat. Was passiert mit dem von Ihnen konfigurierten Löschverhalten, und was wäre mit Cascade passiert?
- ✓ Zwei Agents laden das Ticket SD-1042 und klicken im Abstand von einer Sekunde auf Save. Welches Speichern löst eine Exception aus, was steht in der WHERE-Klausel der UPDATE-Anweisung, und was sieht der Benutzer?
- ✓ Warum liefert IsRequired() auf Title dem Benutzer keine Meldung „Enter a title“, und woher kommt diese Meldung später im Kurs?

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.



Dieses Beispiel haben wir schon gebaut – vergleichen Sie Ihres mit unserem

Die Referenzimplementierung zu Modul 2 liegt im öffentlichen Beispiel-Repository. Bauen Sie zuerst Ihre eigene Version und öffnen Sie dann unsere: Die Unterschiede sind der interessante Teil.



[Beispiel zu Modul 2 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie [Data Binding with EF Core Course/Module 2](#).



Unser Beispiel ist ein Ausgangspunkt, kein fertiges Produkt

Es ist genauso entstanden wie Ihres – per Prompt und anschließend korrigiert – und endet bewusst dort, wo das Modul endet. Rechnen Sie mit Platzhaltern, wo Produktivcode hingehört, mit knapper Fehlerbehandlung und ohne Tests. Verstehen Sie das als Einladung: Nutzen Sie, was Sie in diesem Modul gelernt haben, und dazu den Assistenten, um es weiterzuentwickeln. Code zu verbessern, den Sie nicht selbst geschrieben haben, ist die Fähigkeit, die dieser Kurs eigentlich vermittelt.

Drei Prompts für den nächsten Schritt

Richten Sie Ihren Assistenten auf das Projekt, das Sie gerade gebaut haben – oder auf unseres –, und probieren Sie diese Prompts aus. Jeder ist eine Übung zum Prüfen und Erweitern, kein Neuschreiben.

- ✓ **Beweisen Sie, dass es wirklich funktioniert.** *"Geh dieses Projekt Ergebnis für Ergebnis anhand der Modulspezifikation unter <https://learnwisej.com/assets/courses/ef-core-binding/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."*
- ✓ **Prüfen Sie es wie einen Pull-Request.** *"Führ mich durch diesen SupportDesk-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 SupportDesk-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."*



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.