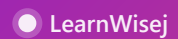


Exercice de code avec l'IA

Construisez l'exemple SupportDesk de ce module avec ChatGPT ou Claude à partir d'un prompt prêt à l'emploi qui oriente le modèle vers la spécification du module, puis relisez, exécutez et étendez ce qu'il vous renvoie — y compris en le comparant à notre propre solution de référence sur GitHub.



L'assistant, ce sont les mains rapides. L'architecte, c'est toujours vous.

Un assistant peut produire l'exemple de ce module en quelques minutes. Ce qu'il ne peut pas faire, c'est décider si le résultat est la bonne conception pour votre application — cette décision vous revient. C'est vous qui apportez le jugement : briefez-le correctement, puis vérifiez ce qu'il vous renvoie à la lumière de ce que le Module 2 vient de vous apprendre.

Ce que vous demandez

Créez les entités Customer, Agent, Category, Ticket et TicketComment de la Support Desk Data Console, configurez les relations et les comportements de suppression avec la Fluent API, configurez RowVersion avec IsRowVersion, ajoutez des index pour Status, DueDate, CustomerId et UpdatedAt, ajoutez une fabrique de conception, créez la migration InitialCreate et insérez cinq clients, trois agents, des catégories et au moins cinquante tickets dans une base locale SQLite ou SQL Server LocalDB.

Le module demande ces livrables, et le prompt ci-dessous demande exactement ceux-là à l'assistant :

- ✓ Entités Customer, Agent, Category, Ticket et TicketComment avec leurs relations et comportements de suppression
- ✓ RowVersion configuré avec IsRowVersion
- ✓ Index pour Status, DueDate, CustomerId et UpdatedAt
- ✓ Fabrique de conception et migration InitialCreate
- ✓ Données initiales: cinq clients, trois agents, des catégories et au moins cinquante tickets

Pourquoi il est conçu ainsi

La Support Desk Data Console affichera les tickets dans une grille paginable, les modifiera dans un formulaire modal et laissera plusieurs agents travailler sur la même file d'attente en même temps, et chaque écran ne vaut que ce que vaut le modèle en dessous. Un filtre sans index correspondant devient un parcours de table à cinquante mille tickets, deux agents qui enregistrent le même ticket perdent silencieusement le travail de l'un d'eux si la ligne ne porte pas de jeton de concurrence, et un schéma corrigé à la main casse le déploiement suivant. Le modèle est le contrat entre l'interface et la base de données; ce module le construit donc explicitement et le versionne comme du code source.

Voici les termes que l'assistant doit maîtriser. Si ce qu'il vous renvoie les emploie de façon approximative, il n'a pas compris le module :

Terme	Ce qu'il signifie ici
entité adaptée à la liaison	Une entité avec une clé prévisible, des champs d'affichage stables, des propriétés de navigation explicites et de petites tables de référence, pour qu'une grille, une ComboBox et un éditeur puissent tous s'y lier.
modèle de vue TicketListItem	Une projection à plat des seules colonnes affichées par la grille, pour que la grille ne reçoive jamais tout le graphe de Ticket et reste indépendante de l'unité de travail d'édition.
configuration par la Fluent API	Des règles de mappage écrites dans OnModelCreating (chaînes obligatoires, longueurs, index, comportement de suppression, précision, valeurs par défaut) pour que les règles de persistance vivent à côté du DbContext.
jeton de concurrence RowVersion	Un tableau d'octets généré par la base et marqué IsRowVersion; EF Core ajoute sa valeur d'origine à la clause WHERE de l'UPDATE, si bien qu'un enregistrement obsolète ne met à jour aucune ligne et lève DbUpdateConcurrencyException.
comportement de suppression	La règle OnDelete d'une relation (Restrict, SetNull, Cascade) qui décide si la suppression d'un client est refusée, désaffecte des tickets ou supprime aussi les commentaires.
migration	Une classe C# accompagnée du snapshot du modèle, qui décrit un changement de schéma relu, commité dans le contrôle de source et appliqué dans l'ordre pour amener n'importe quelle base au niveau du modèle.
fabrique de conception	Une IDesignTimeDbContextFactory qui indique aux outils dotnet ef comment construire le Context sans exécuter le démarrage de l'application Wisej.NET.

Construisez-le — choisissez votre assistant

Les deux boutons ouvrent un nouvel onglet avec le **même prompt prêt à l'emploi** déjà rempli. Le prompt oriente le modèle vers la spécification de ce module — un unique fichier Markdown qui contient la leçon, l'énoncé du lab et les livrables — pour qu'il travaille sur le vrai contenu au lieu de deviner de quoi parle le Module 2. Vous pouvez ouvrir vous-même [module-02.md](#) pour voir exactement ce qu'il lira.



Utiliser ChatGPT

Ouvre [chatgpt.com](#) avec le prompt de ce module déjà chargé. Fonctionne avec GPT-5 ou Codex ; toute offre capable de naviguer sur le web récupérera la spécification.



Utiliser Claude

Ouvre [claude.ai](#) avec le même prompt déjà chargé. Si vous utilisez Claude Code, vous pouvez plutôt le coller dans un terminal ouvert sur un dossier vide.



Envoyez-le tel quel et vous obtenez un projet, pas un extrait

Le prompt se termine en demandant toute la solution sous forme d'un unique `.zip`, avec les étapes pour la compiler et l'exécuter. Téléchargez-le, ouvrez la solution dans Visual Studio, appuyez sur F5 et SupportDesk s'ouvre sur `localhost`. Il vous faut le SDK .NET et le package NuGet `Wisej-4`; tout le reste, le prompt demande à l'assistant de l'écrire lui-même.

Je suis le Module 2 - « Modélisation, configuration du DbContext et migrations » - du cours LearnWisej « Liaison de données avec EF Core », et je veux construire avec toi l'exemple de ce module. Réponds-moi en français.

Commence par lire la spécification du module. C'est un unique fichier Markdown (en anglais) qui contient la leçon complète, l'énoncé du lab et les livrables :

<https://learnwisej.com/assets/courses/ef-core-binding/specs/module-02.md>

Construis ensuite un projet Wisej.NET 4 complet que je puisse ouvrir dans Visual Studio :

1. Projet « SupportDesk », C#, net8.0 ou ultérieur, package NuGet Wisej-4 4.1.x, hôte Kestrel qui appelle `app.UseWisej()`.
2. Construis ce que demande le lab de la spécification : Créez les entités Customer, Agent, Category, Ticket et TicketComment de la Support Desk Data Console, configurez les relations et les comportements de suppression avec la Fluent API, configurez RowVersion avec `IsRowVersion`, ajoutez des index pour Status, DueDate, CustomerId et UpdatedAt, ajoutez une fabrique de conception, créez la migration InitialCreate et insérez cinq clients, trois agents, des catégories et au moins cinquante tickets dans une base locale SQLite ou SQL Server LocalDB.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Entités Customer, Agent, Category, Ticket et TicketComment avec leurs relations et comportements de suppression
 - RowVersion configuré avec `IsRowVersion`
 - Index pour Status, DueDate, CustomerId et UpdatedAt
 - Fabrique de conception et migration InitialCreate
 - Données initiales: cinq clients, trois agents, des catégories et au moins cinquante tickets
4. Suis la spécification partout où elle nomme une classe, une propriété ou un événement Wisej.NET - utilise exactement celui-là. Là où elle laisse le choix, dis-moi quelle API tu as choisie et pourquoi.
5. Garde l'autorité côté serveur : l'état et la validation vivent dans .NET, le navigateur affiche et remonte les informations. Ne déplace jamais une décision dans JavaScript parce que c'est plus simple là-bas.
6. Rends le résultat démontrable : je dois pouvoir l'exécuter et voir le comportement dont traite ce module, y compris ce qui se passe quand quelque chose tourne mal.

Avant d'écrire la moindre ligne de code, reformule avec tes propres mots ce que tu vas construire et où vivra chaque élément, pour que je puisse corriger la conception tant que cela ne coûte pas cher.

Quand le code est terminé, rassemble toute la solution dans un unique fichier .zip téléchargeable et donne-moi les étapes exactes pour la compiler et l'exécuter.

Enfin, réponds à ces questions de la spécification en t'appuyant sur ta propre implémentation :

- Vous supprimez un client qui a encore des tickets ouverts. Que se passe-t-il avec le comportement de suppression que vous avez configuré, et que se serait-il passé avec Cascade?
- Deux agents chargent le ticket SD-1042 et cliquent tous deux sur Save à une seconde d'intervalle. Quel enregistrement lève une exception, que contient la clause WHERE de l'instruction UPDATE, et que voit l'utilisateur?
- Pourquoi `IsRequired()` sur Title ne donne-t-il pas à l'utilisateur un message « Enter a title », et d'où viendra ce message plus loin dans le cours?

Pose-moi des questions sur tout ce qui est ambigu avant de commencer.

Puis relisez ce qui vous revient

Un code qui compile n'est pas encore un code juste. Les questions intéressantes sont des questions de conception : est-ce bien le contrôle Wisej.NET que la tâche exige, suit-il le cycle de vie que le framework vous offre, l'état de référence est-il sur le serveur ou a-t-il été laissé dans le navigateur parce que c'était pratique ? Lisez le résultat en relecteur, pas en client.

Ce module vous donne la checklist. Répondez à chaque point en vous appuyant sur le code que vous venez de recevoir :

- ✓ Vous supprimez un client qui a encore des tickets ouverts. Que se passe-t-il avec le comportement de suppression que vous avez configuré, et que se serait-il passé avec Cascade?
- ✓ Deux agents chargent le ticket SD-1042 et cliquent tous deux sur Save à une seconde d'intervalle. Quel enregistrement lève une exception, que contient la clause WHERE de l'instruction UPDATE, et que voit l'utilisateur?
- ✓ Pourquoi IsRequired() sur Title ne donne-t-il pas à l'utilisateur un message « Enter a title », et d'où viendra ce message plus loin dans le cours?

Tout ce que vous avez dû corriger mérite d'être noté. Une courte note des conventions et des décisions retenues, collée dans le premier message de votre prochaine session, vaut plus que n'importe quel prompt astucieux.



Nous l'avons déjà construit — comparez votre version à la nôtre

L'implémentation de référence du Module 2 se trouve dans le dépôt public des exemples. Construisez d'abord la vôtre, puis ouvrez la nôtre : ce sont les différences qui sont intéressantes.



[Ouvrir l'exemple du Module 2 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Data Binding with EF Core Course/Module 2`.



Notre exemple est un point de départ, pas un produit fini

Il a été construit de la même manière que vous construisez le vôtre — par prompt, puis corrigé — et il s'arrête volontairement là où s'arrête le module. Attendez-vous à des éléments provisoires là où irait du code de production, à une gestion des erreurs minimale et à l'absence de tests. Voyez-y une invitation : utilisez ce que vous avez appris dans ce module, avec l'assistant, pour aller plus loin. Améliorer du code que vous n'avez pas écrit, c'est précisément la compétence que ce cours enseigne.

Trois prompts pour aller plus loin

Orientez votre assistant vers le projet que vous venez de construire — ou vers le nôtre et essayez ceux-ci. Chacun est un exercice de relecture et d'extension, pas une réécriture.

- ✓ **Prouvez que ça fonctionne vraiment.** *"Passe en revue ce projet livrable par livrable par rapport à la spécification du module sur <https://learnwisej.com/assets/courses/ef-core-binding/specs/module-02.md> et montre-moi la preuve que chacun est réellement terminé — le chemin dans le code, où cliquer et ce que je dois voir. Là où un livrable n'est que partiellement satisfait, dis-le clairement au lieu de le défendre. Ensuite, ajoute ce qui manque."*
- ✓ **Relisez-le comme une pull request.** *"Fais-moi parcourir ce code SupportDesk comme si tu relisais la pull request d'un collègue et que tu ne l'avais pas écrit. Où vit l'état, que se passe-t-il de bout en bout lors d'une action de l'utilisateur, que renverrais-tu pour correction, et quelles parties ne serais-tu pas à l'aise de montrer à un client ? Sois précis et cite les lignes dont tu parles — ne me résume pas la conception."*
- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet SupportDesk qui remplace le plus manifestement quelque chose de réel — une source de données factice, un stockage en mémoire, un service simplifié — et remplace-la par une implémentation digne de la production. Le contrat de part et d'autre ne doit pas changer : mêmes types, mêmes appels, même comportement en cas d'échec. Dis-moi ce que l'implémentation réelle t'a obligé à gérer et que la version provisoire ne gérait pas."*



Emportez-le dans le lab

Le lab pratique qui clôt ce module demande les mêmes livrables que ceux que vous venez de faire produire à l'assistant. Servez-vous de ce que vous avez ici comme forme de votre réponse et comme preuve à l'appui — pas comme la réponse elle-même. Le lab est noté sur votre raisonnement, et cette partie doit venir de vous.