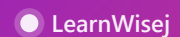


Exercice de code avec l'IA

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 7 vient de vous apprendre.

Ce que vous demandez

Rendez la Support Desk Data Console sûre pour plusieurs utilisateurs : ajoutez RowVersion au modèle d'édition et à son état caché, définissez OriginalValue avant d'enregistrer une entité existante, reproduisez un conflit en modifiant le même ticket dans deux sessions de navigateur, interceptez DbUpdateConcurrencyException, construisez une liste de conflits avec les valeurs proposées et les valeurs de la base, ajoutez les chemins Reload et Overwrite selon la politique, générez un script de migration SQL idempotent avec des notes de déploiement et des chaînes de connexion propres à chaque environnement, laissez la journalisation des données sensibles désactivée hors développement, puis réalisez la revue du projet de synthèse.

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

- ✓ RowVersion transporté dans le modèle d'édition et défini comme OriginalValue avant l'enregistrement
- ✓ Conflit de concurrence reproductible intercepté comme DbUpdateConcurrencyException
- ✓ Boîte de dialogue de conflit listant les valeurs proposées et celles de la base avec les chemins Reload et Overwrite
- ✓ Script de migration SQL idempotent avec notes de déploiement et chaînes de connexion propres à chaque environnement
- ✓ Journalisation des données sensibles désactivée hors développement et revue du projet de synthèse réalisée

Pourquoi il est conçu ainsi

Deux opérateurs ouvrent le ticket T-1042 dans deux sessions de navigateur; l'un le ferme, l'autre relève sa priorité, et c'est l'enregistrement qui arrive en second qui décide de ce dont la base se souvient. Verrouiller

les lignes n'est pas envisageable dans une application web; la Support Desk Data Console doit donc laisser les deux utilisateurs travailler, détecter la collision au moment de l'enregistrement, l'expliquer et laisser une personne décider. Ce module couvre aussi les transactions, la configuration, les migrations, la journalisation, les tests et le projet de synthèse.

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
concurrency optimiste	Les utilisateurs modifient librement; au lieu de verrouiller les lignes, l'application détecte au moment de l'enregistrement que quelqu'un d'autre a modifié la ligne depuis son chargement.
jeton RowVersion	Un jeton de concurrence généré par la base; EF Core place sa valeur d'origine dans la clause WHERE de l'UPDATE, si bien qu'une ligne modifiée entre-temps ne met à jour aucune ligne.
OriginalValue	La version chargée par l'éditeur, restaurée sur l'entité suivie avant SaveChangesAsync pour qu'EF Core compare avec ce que l'utilisateur a vu.
DbUpdateConcurrencyException	Levée quand une ligne attendue n'a pas été touchée; ses Entries contiennent les valeurs proposées, et GetDatabaseValuesAsync récupère ce que la base contient maintenant.
boîte de dialogue de conflit	Un formulaire Wisej.NET qui liste le champ, votre valeur, la valeur en base et la valeur d'origine, et propose Reload, Overwrite quand la politique le permet, ou Merge.
stratégie d'exécution	Le mécanisme de relance d'EF Core pour les échecs transitoires; une transaction explicite doit s'exécuter à l'intérieur pour qu'une relance puisse rejouer toute l'unité de travail.
script de migration idempotent	Le SQL produit par dotnet ef migrations script --idempotent, qui vérifie l'historique des migrations avant chaque étape et peut donc s'exécuter sans risque sur n'importe quel environnement.

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 7. Vous pouvez ouvrir vous-même [module-07.md](#) pour voir exactement ce qu'il lira.



Utiliser ChatGPT

Ouvre [chatgpt.com](#) avec le prompt de ce module déjà chargé. Fonctionne avec GPT-



Utiliser Claude

Ouvre [claude.ai](#) avec le même prompt déjà chargé. Si vous utilisez Claude Code,

5 ou Codex ; toute offre capable de naviguer sur le web récupérera la spécification.

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 7 – « Concurrency, transactions, deployment, diagnostic and project of synthesis » – 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-07.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 : Rendez la Support Desk Data Console sûre pour plusieurs utilisateurs : ajoutez RowVersion au modèle d'édition et à son état caché, définissez OriginalValue avant d'enregistrer une entité existante, reproduisez un conflit en modifiant le même ticket dans deux sessions de navigateur, interceptez `DbUpdateConcurrencyException`, construisez une liste de conflits avec les valeurs proposées et les valeurs de la base, ajoutez les chemins Reload et Overwrite selon la politique, générez un script de migration SQL idempotent avec des notes de déploiement et des chaînes de connexion propres à chaque environnement, laissez la journalisation des données sensibles désactivée hors développement, puis réalisez la revue du projet de synthèse.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - RowVersion transporté dans le modèle d'édition et défini comme OriginalValue avant l'enregistrement
 - Conflit de concurrence reproductible intercepté comme `DbUpdateConcurrencyException`
 - Boîte de dialogue de conflit listant les valeurs proposées et celles de la base avec les chemins Reload et Overwrite
 - Script de migration SQL idempotent avec notes de déploiement et chaînes de connexion propres à chaque environnement
 - Journalisation des données sensibles désactivée hors développement et revue du projet de synthèse réalisée
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 :

- Si vous omettiez la ligne OriginalValue, que montrerait le test à deux sessions, et pourquoi aurait-il l'air d'une réussite tout en perdant silencieusement la modification du premier opérateur?

– Quels chemins de résolution proposeriez-vous à un agent ordinaire et lesquels à un superviseur, et que doit-il arriver à la grille et au RowVersion du modèle d'édition après chaque chemin?

– Trois instances de production démarrent après un déploiement. Dans votre conception, laquelle applique la migration et quand, et qu'est-ce qui tournerait mal si Migrate() s'exécutait au démarrage de chaque instance?

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 :

- ✓ Si vous omettiez la ligne OriginalValue, que montrerait le test à deux sessions, et pourquoi aurait-il l'air d'une réussite tout en perdant silencieusement la modification du premier opérateur?
- ✓ Quels chemins de résolution proposeriez-vous à un agent ordinaire et lesquels à un superviseur, et que doit-il arriver à la grille et au RowVersion du modèle d'édition après chaque chemin?
- ✓ Trois instances de production démarrent après un déploiement. Dans votre conception, laquelle applique la migration et quand, et qu'est-ce qui tournerait mal si Migrate() s'exécutait au démarrage de chaque instance?

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 7 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 7 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Data Binding with EF Core Course/Module 7`.



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-07.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.