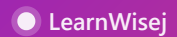


Exercice de code avec l'IA

Construisez l'exemple EnterpriseOps 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.



Vous ne taperez pas cet exemple. Vous le piloterez.

Presque plus personne n'écrit du code Wisej.NET à partir d'un fichier vide — un assistant rédige le premier jet et le développeur décide s'il est correct. Cela ne fonctionne que si vous maîtrisez le sujet, et c'est ce que le reste de ce module vous a apporté. Ici, vous confiez ces connaissances à ChatGPT ou à Claude, vous récupérez l'exemple du module sous forme de solution Visual Studio fonctionnelle, et vous le relisez comme vous reliriez la pull request d'un collègue.

Ce que vous demandez

Construisez un Import Center avec une file de jobs, un observateur de progression, l'annulation, un traitement des lignes pouvant être relancé, des notifications et un écran de détail des jobs. Démontrez la reprise après la fermeture et la réouverture de la session.

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

- ✓ Modèle de job et magasin de statuts
- ✓ Abstraction de file d'attente en arrière-plan
- ✓ UI d'observation de la progression
- ✓ Panneau de notifications
- ✓ Politique de nouvelles tentatives et d'annulation

Pourquoi il est conçu ainsi

Les systèmes Wisej.NET avancés exécutent souvent des imports, de la génération de rapports, des intégrations et des notifications pendant que les utilisateurs continuent de travailler. Ces pipelines doivent être observables, annulables et sûrs d'une session à l'autre.

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
modèle de job	Enregistrement persistant décrivant une opération en arrière-plan: son type, ses entrées, son statut, sa progression, son propriétaire et son résultat, indépendamment de tout écran ouvert.
file d'attente en arrière-plan	File d'attente durable qui accepte des jobs et les traite en dehors du cycle de requête, pour qu'un travail de longue durée ne bloque jamais l'interface utilisateur.
Application.StartTask	L'API Wisej.NET qui exécute du travail sur un thread en arrière-plan rattaché au contexte de session, pour pouvoir mettre à jour l'UI en toute sécurité une fois terminé.
Application.Update	Pousse immédiatement vers le navigateur les modifications d'UI en attente effectuées depuis un thread en arrière-plan; c'est ainsi que la progression et les notifications apparaissent sans action de l'utilisateur.
service de notifications	Service central qui remet les résultats des jobs et les alertes à la bonne session utilisateur, que l'écran d'origine soit encore ouvert ou non.
politique de nouvelles tentatives	Règles définissant combien de fois et à quel rythme un job en échec est relancé, avec un backoff, pour que les erreurs transitoires se résorbent sans surcharger les systèmes en aval.
idempotence	Concevoir les jobs de sorte que les exécuter deux fois produise le même résultat qu'une seule exécution, ce qui rend sûrs les nouvelles tentatives et les messages en double.
audit des jobs	Trace de qui a lancé chaque job, quand il s'est exécuté et ce qu'il a modifié, pour que le traitement en arrière-plan soit aussi traçable que le travail interactif.

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 6. Vous pouvez ouvrir vous-même [module-06.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 EnterpriseOps 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 6 - « Systèmes temps réel, pipelines en arrière-plan, notifications et imports » - du cours LearnWisej « Wisej.NET en entreprise : de l'architecture au cloud », 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/azure-docker/specs/module-06.md>

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

1. Projet « EnterpriseOps », 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 : Construisez un Import Center avec une file de jobs, un observateur de progression, l'annulation, un traitement des lignes pouvant être relancé, des notifications et un écran de détail des jobs. Démontrez la reprise après la fermeture et la réouverture de la session.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Modèle de job et magasin de statuts
 - Abstraction de file d'attente en arrière-plan
 - UI d'observation de la progression
 - Panneau de notifications
 - Politique de nouvelles tentatives et d'annulation
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 :

- À qui appartient le job après la fermeture de la session?
- À quelle fréquence l'UI se met-elle à jour?
- Quelles erreurs peuvent être relancées et lesquelles sont définitives?

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 :

- ✓ À qui appartient le job après la fermeture de la session?
- ✓ À quelle fréquence l'UI se met-elle à jour?
- ✓ Quelles erreurs peuvent être relancées et lesquelles sont définitives?

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 6 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 6 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Enterprise Wisej.NET Course/Module 6`.



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.

- ✓ **Relisez-le comme une pull request.** *"Fais-moi parcourir ce code EnterpriseOps 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 EnterpriseOps 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 gèrerait pas."*
- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet EnterpriseOps pour repérer les modes de défaillance qu'il ne gère pas encore — une saisie erronée ou manquante, une dépendance lente ou indisponible, un traitement qui aboutit après que l'utilisateur a changé de page, et deux actions qui se chevauchent. Pour chacun, dis-moi d'abord s'il s'agit d'un risque réel dans Wisej.NET 4 et pourquoi, et seulement ensuite écris la correction. Là où tu estimes qu'aucune gestion n'est nécessaire, dis-le et explique ton raisonnement au lieu d'ajouter du code défensif."*



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.