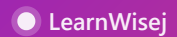


Exercice de code avec l'IA

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.



Briefez-le, lisez-le, puis cassez-le.

Cet exercice comporte trois parties, et seule la première concerne le prompt. Vous briefez un assistant avec la spécification du module et récupérez une solution EnterpriseOps ; vous lisez ce qu'il a écrit et décidez s'il fait vraiment ce que le module demandait ; puis vous le poussez plus loin que ce que le module demandait. C'est dans la deuxième et la troisième partie que l'on apprend.

Ce que vous demandez

Concevez et prototypiez un mode Field Technician pour EnterpriseOps. Ajoutez une mise en page adaptée à l'appareil, un cache hors ligne des ordres de travail, une file locale des clôtures, une simulation de synchronisation à la reconnexion et l'affichage des conflits.

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

- ✓ Note d'architecture Hybrid/hors ligne
- ✓ Abstraction des services de l'appareil
- ✓ Modèle de file hors ligne
- ✓ Écran des conflits de synchronisation
- ✓ Checklist d'utilisabilité du mode terrain

Pourquoi il est conçu ainsi

Certains workflows d'entreprise exigent un fonctionnement sur mobile, tablette, terrain, borne ou hors ligne. Les développeurs Wisej.NET avancés doivent décider quand les schémas Hybrid/PWA sont pertinents et comment préserver un modèle d'application unique sans ignorer les réalités des appareils.

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
Wisej.NET Hybrid	La prise en charge par le framework de l'exécution de la même application dans le navigateur, en tant qu'application de bureau ou mobile, et avec un accès natif à l'appareil.
mode connecté	Le modèle standard où l'application s'exécute sur le serveur et où l'appareil est un client. C'est le plus simple, mais il nécessite une connexion réseau.
application hors ligne/locale	Exécuter l'application localement sur l'appareil pour qu'elle continue de fonctionner sans connexion, au prix de la gestion des données locales et de la synchronisation.
abstraction de l'appareil	Placer la caméra, le système de fichiers, le GPS et les autres fonctionnalités de l'appareil derrière des interfaces de service, pour que les écrans ne dépendent pas d'une plateforme précise.
SQLite local	Base de données embarquée légère sur l'appareil, qui stocke les données de travail hors ligne et alimente la file de synchronisation.
file de synchronisation	Liste durable des modifications locales en attente d'envoi au serveur au retour de la connectivité, traitées dans l'ordre et relancées sans risque.
résolution des conflits	Règles qui déterminent ce qui se passe quand le même enregistrement a été modifié à la fois localement et sur le serveur pendant la période hors ligne, décidées par le produit et non par le hasard.
shell PWA	Le manifeste de progressive web app et le service worker qui permettent d'installer l'application sur un appareil et de charger son shell sans réseau.

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 13. Vous pouvez ouvrir vous-même [module-13.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 13 – « Applications Wisej.NET hybrides, PWA, hors ligne et adaptées à l'appareil » – 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-13.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 : Concevez et prototypez un mode Field Technician pour EnterpriseOps. Ajoutez une mise en page adaptée à l'appareil, un cache hors ligne des ordres de travail, une file locale des clôtures, une simulation de synchronisation à la reconnexion et l'affichage des conflits.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Note d'architecture Hybrid/hors ligne
 - Abstraction des services de l'appareil
 - Modèle de file hors ligne
 - Écran des conflits de synchronisation
 - Checklist d'utilisabilité du mode terrain
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 :

- Quelles données sont autorisées hors ligne?
- Comment les permissions sont-elles rafraîchies?
- Que se passe-t-il quand des modifications locales entrent en conflit avec celles du serveur?

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 :

- ✓ Quelles données sont autorisées hors ligne?
- ✓ Comment les permissions sont-elles rafraîchies?
- ✓ Que se passe-t-il quand des modifications locales entrent en conflit avec celles du serveur?

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 13 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 13 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Enterprise Wisej.NET Course/Module 13`.



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.

- ✓ **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."*
- ✓ **Ajoutez le deuxième cas.** *"Étends ce projet EnterpriseOps avec un deuxième cas qui suit le même modèle que le premier — une autre entité, un autre écran ou un autre contrôle de même forme, câblé de la même manière. Extrais dans une classe de base ou un service ce qui est réellement partagé, mais seulement là où cela se justifie ; deux choses similaires n'appellent pas toujours une abstraction. Montre-moi le diff et justifie chaque élément que tu as extrait."*
- ✓ **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/azure-docker/specs/module-13.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."*



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.