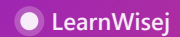


Exercice de code avec l'IA

Construisez l'exemple FieldOps 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.



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

Créez la solution FieldOps avec une liste des ordres de travail, une page de détail d'ordre de travail et un flux de mise à jour du statut qui fonctionnent dans un simple navigateur. Rédigez le dossier de décision de distribution : comparez le navigateur, la PWA, le shell desktop et le shell mobile au regard des besoins des techniciens, choisissez les cibles et leur ordre, et définissez les vérifications de capacités que l'application utilisera pour activer les fonctionnalités de l'appareil.

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

- ✓ Solution FieldOps avec liste, détail et mise à jour du statut
- ✓ Tableau comparatif des modèles de distribution
- ✓ Dossier de décision de distribution avec l'ordre des cibles
- ✓ Conception de la vérification des capacités
- ✓ Schéma d'architecture avec un serveur et plusieurs shells

Pourquoi il est conçu ainsi

FieldOps est une application d'intervention sur le terrain: les techniciens reçoivent des ordres de travail, capturent des photos et des signatures, se rendent sur les sites et travaillent avec une connectivité intermittente. Construite comme une application Wisej.NET ordinaire, elle fonctionne déjà dans n'importe quel navigateur, ce qui suffit aux répartiteurs du bureau, mais pas à un technicien debout dans une salle des machines avec une seule barre de réseau et un téléphone dans une main gantée. La question à laquelle répond ce module n'est pas « pouvons-nous la construire » mais « comment atteint-elle les personnes qui l'utilisent » — et cette réponse est une décision d'architecture, pas une considération de packaging après coup.

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 distribution	Le chemin que prend l'application jusqu'à l'utilisateur: un onglet de navigateur, une PWA installée, une fenêtre desktop ou une application mobile distribuée via un store. Il change la distribution, les mises à jour et l'accès à l'appareil — jamais le C# qui construit l'écran.
moteur de navigateur hôte	Chaque modèle affiche la même UI pilotée par le serveur dans un moteur de navigateur: le navigateur de l'utilisateur ou un moteur embarqué — WebView2 sous Windows, WebKit sous macOS et iOS, la WebView système sous Android.
Wisej.NET Hybrid	Le shell du framework basé sur .NET MAUI, qui héberge votre application Wisej.NET existante sur Android, iOS et desktop et ajoute un bridge vers les fonctionnalités de l'appareil comme la caméra, la géolocalisation, les fichiers et les notifications.
bridge vers l'appareil	Le canal bidirectionnel entre votre C# côté serveur et le shell natif: le serveur demande une action sur l'appareil, le shell l'exécute et le résultat revient à votre code sous forme d'événement, pas de valeur de retour.
vérification de capacités	Un test à l'exécution de ce que le shell courant sait réellement faire, utilisé pour afficher, masquer ou remplacer une fonctionnalité afin qu'un même écran se comporte correctement dans chaque modèle de distribution.
cache du shell de l'application	Le HTML, le CSS, les scripts et les icônes statiques qu'un service worker de PWA peut conserver localement. C'est ce qui se charge encore sans réseau — la partie interactive de l'application, rendue par le serveur, n'en fait pas partie.
asymétrie des mises à jour	Le serveur peut être redéployé à tout moment et chaque session obtient le nouveau build au rechargement, alors qu'un installateur desktop ou un package store exige d'abord une release puis une installation.

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 1. Vous pouvez ouvrir vous-même [module-01.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 FieldOps 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 1 – « Modèles d'hébergement au-delà du navigateur » – du cours LearnWisej « Packaging hybride et natif », 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/hybrid-native-packaging/specs/module-01.md>

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

1. Projet « FieldOps », 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 la solution FieldOps avec une liste des ordres de travail, une page de détail d'ordre de travail et un flux de mise à jour du statut qui fonctionnent dans un simple navigateur. Rédigez le dossier de décision de distribution : comparez le navigateur, la PWA, le shell desktop et le shell mobile au regard des besoins des techniciens, choisissez les cibles et leur ordre, et définissez les vérifications de capacités que l'application utilisera pour activer les fonctionnalités de l'appareil.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Solution FieldOps avec liste, détail et mise à jour du statut
 - Tableau comparatif des modèles de distribution
 - Dossier de décision de distribution avec l'ordre des cibles
 - Conception de la vérification des capacités
 - Schéma d'architecture avec un serveur et plusieurs shells
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 :

- Un technicien perd le signal au milieu de la mise à jour d'un ordre de travail. Décrivez précisément quelles parties de FieldOps peuvent encore être affichées depuis un shell en cache, lesquelles ne le peuvent pas, et ce que votre écran doit faire au moment où la connexion tombe.
- Votre dossier de décision place le shell Android avant la PWA. Qu'est-ce qui devrait être vrai des appareils et du réseau des techniciens pour que cet ordre soit mauvais?
- La page de détail a besoin d'une photo. Expliquez pourquoi une vérification de capacités associée à un repli Upload est une meilleure conception qu'une option de compilation qui produit une version de l'écran réservée au mobile.

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 :

- ✓ Un technicien perd le signal au milieu de la mise à jour d'un ordre de travail. Décrivez précisément quelles parties de FieldOps peuvent encore être affichées depuis un shell en cache, lesquelles ne le peuvent pas, et ce que votre écran doit faire au moment où la connexion tombe.
- ✓ Votre dossier de décision place le shell Android avant la PWA. Qu'est-ce qui devrait être vrai des appareils et du réseau des techniciens pour que cet ordre soit mauvais?
- ✓ La page de détail a besoin d'une photo. Expliquez pourquoi une vérification de capacités associée à un repli Upload est une meilleure conception qu'une option de compilation qui produit une version de l'écran réservée au mobile.

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.



Pas encore d'exemple de référence pour ce cours — le vôtre est le premier

Le [dépôt des exemples LearnWisej](#) contient des solutions complètes pour la plupart des cours, mais pas encore pour celui-ci. Il n'y a donc rien à quoi vous comparer et aucun raccourci : la spécification et votre propre relecture sont votre seul filet de sécurité. Gardez ce que vous construisez — et si le résultat est réussi, nous aimerions le voir.

Trois prompts pour aller plus loin

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

- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet FieldOps 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 gèrait pas."*
- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet FieldOps 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 FieldOps 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."*



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.