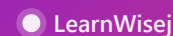


Exercice de code avec l'IA

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.



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 FieldOps ; 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

Packagez FieldOps pour Windows : un shell WebView2 avec une fenêtre native, un titre personnalisé et une icône dans la zone de notification, un bridge qui permet à l'application d'ouvrir un sélecteur de dossier natif pour exporter les rapports d'ordres de travail et d'afficher une notification native à l'arrivée d'un nouvel ordre, une configuration qui dirige le shell vers le serveur distant avec un repli local, et un installeur avec une vérification des mises à jour au lancement.

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

- ✓ Shell desktop WebView2 avec fenêtre et icône de notification
- ✓ Bridge natif pour le sélecteur de dossier et les notifications
- ✓ Configuration du serveur distant et local
- ✓ Package d'installation
- ✓ Vérification des mises à jour au lancement

Pourquoi il est conçu ainsi

Les techniciens de FieldOps passent toute leur journée dans l'application: un portable dans la camionnette, une station d'accueil au dépôt. Le build navigateur des deux premiers modules fait le travail, mais il n'a pas d'icône dans le menu Démarrer, aucune présence dans la zone de notification quand une intervention est attribuée, aucun moyen de déposer un rapport exporté dans un dossier choisi par le technicien, et rien que le service informatique puisse déployer par son canal logiciel habituel. Ce module enveloppe cette même application, sans la modifier, dans un shell Windows qui apporte exactement ces quatre éléments.

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
contrôle WebView2	Le moteur Chromium que Microsoft distribue sous la forme de l'Evergreen Runtime redistribuable, intégré dans une fenêtre WinForms ou WPF pour que votre page s'affiche exactement comme dans Edge.
shell desktop	Un petit exécutable natif qui possède la fenêtre, l'icône de notification et les appels au système, et qui pointe un contrôle WebView2 vers votre application Wisej.NET.
chrome de fenêtre	Le titre, l'icône, la taille minimale, la position mémorisée et le comportement en instance unique qui donnent à l'application l'allure d'un logiciel installé plutôt qu'embarqué.
bridge de messages	Le canal bidirectionnel entre la page et le shell: <code>PostWebMessageAsJson</code> vers la page, <code>WebMessageReceived</code> vers le shell, ou un objet hôte publié avec <code>AddHostObjectToScript</code> .
interopérabilité JavaScript	La façon dont le C# côté serveur atteint ce canal: <code>Application.Eval</code> et <code>Application.Call</code> exécutent du script dans le contexte du navigateur de la session, et la réponse revient sous forme d'événement client.
topologie du serveur	Le fait que le shell navigue vers un serveur distant de l'entreprise ou démarre un processus Wisej.NET auto-hébergé sur le loopback — décidé au lancement et journalisé, jamais deviné.
canal de mise à jour	Un endpoint de version que le shell interroge au démarrage, plus l'installateur qu'il récupère quand la réponse est plus récente que la version qu'il exécute.

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 3. Vous pouvez ouvrir vous-même [module-03.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 3 - « Packaging desktop avec un shell WebView » - 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-03.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 : Packagez FieldOps pour Windows : un shell WebView2 avec une fenêtre native, un titre personnalisé et une icône dans la zone de notification, un bridge qui permet à l'application d'ouvrir un sélecteur de dossier natif pour exporter les rapports d'ordres de travail et d'afficher une notification native à l'arrivée d'un nouvel ordre, une configuration qui dirige le shell vers le serveur distant avec un repli local, et un installeur avec une vérification des mises à jour au lancement.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Shell desktop WebView2 avec fenêtre et icône de notification
 - Bridge natif pour le sélecteur de dossier et les notifications
 - Configuration du serveur distant et local
 - Package d'installation
 - Vérification des mises à jour au lancement
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 lance deux exports à moins d'une seconde d'intervalle et le shell répond aux deux boîtes de dialogue de dossier. Qu'est-ce qui, dans la conception de votre bridge, empêche le second chemin d'être écrit dans le premier rapport, et que se passerait-il sans cela?
- Votre bouton Export ouvre désormais une boîte de dialogue native de sélection de dossier. Décrivez précisément ce que fait ce même bouton lorsque la page est ouverte dans Edge plutôt que dans le shell, et à quel endroit de votre code cette différence est décidée.
- L'endpoint de version est injoignable parce que la camionnette n'a pas de réseau. Argumentez sur ce que le shell doit faire lors de ce lancement, et expliquez ce qu'un technicien perdrait avec chacune des alternatives que vous avez écartées.

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 lance deux exports à moins d'une seconde d'intervalle et le shell répond aux deux boîtes de dialogue de dossier. Qu'est-ce qui, dans la conception de votre bridge, empêche le second chemin d'être écrit dans le premier rapport, et que se passerait-il sans cela?
- ✓ Votre bouton Export ouvre désormais une boîte de dialogue native de sélection de dossier. Décrivez précisément ce que fait ce même bouton lorsque la page est ouverte dans Edge plutôt que dans le shell, et à quel endroit de votre code cette différence est décidée.
- ✓ L'endpoint de version est injoignable parce que la camionnette n'a pas de réseau. Argumentez sur ce que le shell doit faire lors de ce lancement, et expliquez ce qu'un technicien perdrait avec chacune des alternatives que vous avez écartées.

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.

- ✓ **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."*
- ✓ **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/hybrid-native-packaging/specs/module-03.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 FieldOps 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."*



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.