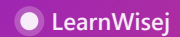


Exercice de code avec l'IA

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.



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 2 vient de vous apprendre.

Ce que vous demandez

Transformez FieldOps en PWA : ajoutez le manifeste avec les icônes, la couleur de thème et l'affichage standalone, enregistrez un service worker qui met en cache le shell et les ressources statiques, affichez une invitation à l'installation après la première session réussie, affichez une page hors ligne avec le dernier récapitulatif synchronisé des ordres de travail lorsque le serveur est injoignable, et implémentez le flux de mise à jour qui recharge l'application quand une nouvelle version est disponible.

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

- ✓ Manifeste d'application web avec icônes et couleurs
- ✓ Service worker mettant en cache le shell
- ✓ Invitation à l'installation après la première session
- ✓ Page hors ligne avec le dernier récapitulatif synchronisé
- ✓ Flux de mise à jour et vérification de version

Pourquoi il est conçu ainsi

Les techniciens de FieldOps n'ouvrent pas un navigateur pour taper une URL. Ils touchent une icône sur un téléphone resté toute la matinée dans une camionnette, dans une salle des machines où le signal tombe à zéro au milieu d'une intervention. Une progressive web app leur donne cette icône, une fenêtre sans chrome de navigateur et une page qui dit encore quelque chose d'utile quand le réseau disparaît — mais FieldOps est une application Wisej.NET, donc son interface utilisateur vit sur le serveur, et faire comme si ce n'était pas le cas, c'est ainsi qu'une PWA se transforme en ticket de support.

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
manifeste d'application web	Un petit fichier JSON, référencé depuis la page de démarrage, qui indique au navigateur le nom de l'application, ses icônes, ses couleurs, son mode d'affichage et son URL de démarrage — sans lui, le navigateur n'a rien à installer.
mode d'affichage	Le membre display du manifeste. standalone lance FieldOps dans sa propre fenêtre sans barre d'adresse, ce qui fait qu'une application installée ne ressemble plus à un simple favori.
icônes maskable	Des icônes déclarées avec purpose: "maskable" et une marge de sécurité, pour qu'Android puisse les découper selon sa propre forme sans rogner le visuel.
service worker	Un script que le navigateur exécute à côté de la page, servi depuis l'origine même de l'application en HTTPS, et qui peut répondre aux requêtes de la page depuis un cache quand la connexion est lente ou absente.
shell de l'application	Les fichiers statiques dont une session a besoin avant que le serveur puisse dire quoi que ce soit: la page de démarrage, les scripts client, le thème et les icônes. C'est ce que le worker est autorisé à mettre en cache.
cache d'abord ou réseau uniquement	La règle de routage du gestionnaire fetch du worker: les fichiers statiques du shell viennent d'abord du cache, tandis que tout ce que la session Wisej.NET échange avec le serveur doit aller directement au réseau, jamais à une copie en cache.
cycle de vie des mises à jour	Un nouveau worker s'installe pendant que l'ancien contrôle encore la page, puis il attend. skipWaiting, clients.claim et un rechargement maîtrisé sont ce qui transforme ce worker en attente en version réellement utilisée par le technicien.

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 2. Vous pouvez ouvrir vous-même [module-02.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 2 - « Mise en place d'une Progressive Web App » - 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-02.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 : Transformez FieldOps en PWA : ajoutez le manifeste avec les icônes, la couleur de thème et l'affichage standalone, enregistrez un service worker qui met en cache le shell et les ressources statiques, affichez une invitation à l'installation après la première session réussie, affichez une page hors ligne avec le dernier récapitulatif synchronisé des ordres de travail lorsque le serveur est injoignable, et implémentez le flux de mise à jour qui recharge l'application quand une nouvelle version est disponible.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Manifeste d'application web avec icônes et couleurs
 - Service worker mettant en cache le shell
 - Invitation à l'installation après la première session
 - Page hors ligne avec le dernier récapitulatif synchronisé
 - Flux de mise à jour et vérification de version
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 relecteur demande pourquoi le service worker ne met pas simplement en cache chaque réponse pour que FieldOps continue de fonctionner en sous-sol. Expliquez ce qu'une réponse de session Wisej.NET rejouée ferait réellement à l'écran du technicien, et où se situe la limite honnête.
- Votre offre d'installation fonctionne sur votre poste mais n'apparaît jamais sur le téléphone du technicien. Nommez trois conditions distinctes qui pourraient en être la cause, et indiquez comment vous les distingueriez depuis l'appareil.
- Un technicien a l'application installée et ouverte lorsque vous déployez un nouveau build. Décrivez chaque étape entre votre déploiement et l'exécution du nouveau shell sur ce

téléphone, et identifiez l'étape qui échouerait silencieusement si le nom du cache ne changeait jamais.

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 relecteur demande pourquoi le service worker ne met pas simplement en cache chaque réponse pour que FieldOps continue de fonctionner en sous-sol. Expliquez ce qu'une réponse de session Wisej.NET rejouée ferait réellement à l'écran du technicien, et où se situe la limite honnête.
- ✓ Votre offre d'installation fonctionne sur votre poste mais n'apparaît jamais sur le téléphone du technicien. Nommez trois conditions distinctes qui pourraient en être la cause, et indiquez comment vous les distingueriez depuis l'appareil.
- ✓ Un technicien a l'application installée et ouverte lorsque vous déployez un nouveau build. Décrivez chaque étape entre votre déploiement et l'exécution du nouveau shell sur ce téléphone, et identifiez l'étape qui échouerait silencieusement si le nom du cache ne changeait jamais.

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.

- ✓ **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."*
- ✓ **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-02.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.