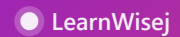


Exercice de code avec l'IA

Construisez l'exemple GlobalDesk 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 7 vient de vous apprendre.

Ce que vous demandez

Terminez le projet de synthèse GlobalDesk. Encapsulez toute recherche de ressource dans un LocalizationService doté de méthodes Text, Date, Number et Currency, donnez-lui une politique de clé manquante qui affiche la clé entre crochets en développement et la journalise en production, et faites de ce service le seul endroit où l'interface touche à ResourceManager. Gardez le métier en dehors de tout cela : le modèle de ticket expose une valeur TicketStatus et l'interface la convertit en libellé localisé. Déroulez ensuite la matrice de tests de localisation sur l'application terminée, en couvrant la culture neutre, l'allemand, une culture langue-région qui retombe sur le repli, une clé volontairement absente, les dates et les montants mis en forme, un changement à l'exécution, une passe sur les chaînes longues et la culture RTL, en vous servant d'un jeu de ressources pseudo-localisé pour révéler tout ce qui reste codé en dur. Vérifiez que les dossiers de culture survivent à une publication, et rendez la checklist QA comme livrable du projet de synthèse.

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

- ✓ LocalizationService avec des méthodes Text, Date, Number et Currency et une politique documentée pour les clés manquantes
- ✓ Code métier qui renvoie une valeur TicketStatus que l'interface convertit en libellé localisé
- ✓ Jeu de ressources pseudo-localisé utilisé pour trouver toutes les chaînes encore codées en dur
- ✓ Matrice de tests de localisation complétée, couvrant le repli, la mise en forme, le changement à l'exécution et le RTL
- ✓ Sortie publiée vérifiée quant à ses dossiers de culture, plus la checklist QA de localisation de GlobalDesk

Pourquoi il est conçu ainsi

Après six modules, GlobalDesk parle deux langues, mais la mécanique est éparpillée: des ressources de designer sur l'éditeur client, une classe utilitaire au-dessus des fichiers.resx partagés, un sélecteur de

langue, un gestionnaire CultureChanged et une démonstration RTL. Rien n'énonce encore ce qui se passe quand une clé n'a pas de traduction, et personne n'a prouvé que la version allemande tient toujours dans une fenêtre étroite. Un système de localisation de production est prévisible: il sait d'où vient chaque chaîne, quelle culture est active dans cette session, à quoi ressemble une traduction manquante, et ce qu'un changement de langue fait aux écrans ouverts. Ce module transforme ces quatre réponses en un petit service, une matrice de tests écrite et le projet de synthèse terminé.

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
LocalizationService	Une petite classe qui centralise les choix de votre application autour de la localisation.NET au lieu de la remplacer: un seul nom de base de ressource, une seule politique de clé manquante, un seul jeu de méthodes de mise en forme.
Politique de clé manquante	Le comportement choisi et testé quand GetString ne renvoie rien: un marqueur [Key] visible pendant le développement, une clé journalisée plus un texte de repli neutre en production.
Texte produit, système et de diagnostic	Trois familles pour trois propriétaires: vos propres chaînes.resx, les libellés de Wisej.NET via Resources-[LANG].resx, et le texte de journal ou d'exception qui reste dans une seule langue de développement.
Métier sans culture	Le modèle expose une valeur comme TicketStatus.Closed et l'interface la convertit en libellé localisé: la logique métier et ses tests ne dépendent donc jamais d'une langue.
Matrice de tests de localisation	Une liste écrite d'exécutions — neutre, traduite, région qui retombe sur le repli, clé manquante, valeurs mises en forme, changement à l'exécution, chaînes longues, RTL, libellés système — pour que la couverture soit une checklist et non un souvenir.
Pseudo-localisation	Un jeu de ressources de test dont les valeurs sont visiblement altérées et allongées, comme [!!! Save customer record!!!], conçu pour révéler les troncatures et les chaînes codées en dur plutôt que pour traduire quoi que ce soit.
Ressources satellites	.NET compile les ressources spécifiques à une culture en assemblés satellites: les dossiers de culture produits par la compilation doivent donc accompagner l'application lorsque vous la publiez ou l'empaquetez.

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 7. Vous pouvez ouvrir vous-même [module-07.md](#) pour voir exactement ce qu'il lira.



Utiliser ChatGPT



Utiliser Claude

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.

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 GlobalDesk 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 7 - « Architecture de localisation en production, tests et projet de synthèse » - du cours LearnWisej « Localisation dans Wisej.NET », 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/localization-in-wisej-net/specs/module-07.md>

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

1. Projet « GlobalDesk », 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 : Terminez le projet de synthèse GlobalDesk. Encapsulez toute recherche de ressource dans un `LocalizationService` doté de méthodes `Text`, `Date`, `Number` et `Currency`, donnez-lui une politique de clé manquante qui affiche la clé entre crochets en développement et la journalise en production, et faites de ce service le seul endroit où l'interface touche à `ResourceManager`. Gardez le métier en dehors de tout cela : le modèle de ticket expose une valeur `TicketStatus` et l'interface la convertit en libellé localisé. Déroulez ensuite la matrice de tests de localisation sur l'application terminée, en couvrant la culture neutre, l'allemand, une culture langue-région qui retombe sur le repli, une clé volontairement absente, les dates et les montants mis en forme, un changement à l'exécution, une passe sur les chaînes longues et la culture RTL, en vous servant d'un jeu de ressources pseudo-localisé pour révéler tout ce qui reste codé en dur. Vérifiez que les dossiers de culture survivent à une publication, et rendez la checklist QA comme livrable du projet de synthèse.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - `LocalizationService` avec des méthodes `Text`, `Date`, `Number` et `Currency` et une politique documentée pour les clés manquantes
 - Code métier qui renvoie une valeur `TicketStatus` que l'interface convertit en libellé localisé
 - Jeu de ressources pseudo-localisé utilisé pour trouver toutes les chaînes encore codées en dur
 - Matrice de tests de localisation complétée, couvrant le repli, la mise en forme, le changement à l'exécution et le RTL
 - Sortie publiée vérifiée quant à ses dossiers de culture, plus la checklist QA de localisation de GlobalDesk
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 ouvre GlobalDesk en français, une culture que vous n'avez jamais traduite, et voit des libellés anglais avec des dates françaises correctement mises en forme. Expliquez quel mécanisme a produit chaque moitié de cet écran, et dites si quelque chose est cassé.
- Votre exécution pseudo-localisée montre un label qui affiche encore "Save" en anglais courant. Énumérez les trois endroits d'où ce texte pourrait venir, et décrivez comment vous les distingueriez sans deviner.
- Un collègue propose de mettre le CultureInfo résolu en cache dans un champ statique de LocalizationService « parce que le rechercher à chaque appel est coûteux ». Décrivez ce que vivrait un deuxième utilisateur connecté, et ce que vous proposeriez à la place.

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 ouvre GlobalDesk en français, une culture que vous n'avez jamais traduite, et voit des libellés anglais avec des dates françaises correctement mises en forme. Expliquez quel mécanisme a produit chaque moitié de cet écran, et dites si quelque chose est cassé.
- ✓ Votre exécution pseudo-localisée montre un label qui affiche encore "Save" en anglais courant. Énumérez les trois endroits d'où ce texte pourrait venir, et décrivez comment vous les distingueriez sans deviner.
- ✓ Un collègue propose de mettre le CultureInfo résolu en cache dans un champ statique de LocalizationService « parce que le rechercher à chaque appel est coûteux ». Décrivez ce que vivrait un deuxième utilisateur connecté, et ce que vous proposeriez à la place.

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.

- ✓ **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/localization-in-wisej-net/specs/module-07.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 GlobalDesk 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 GlobalDesk 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 gérait pas."*



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.