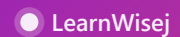


Exercice de code avec l'IA

Construisez l'exemple VisualOperationsStudio 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

Ajoutez à VisualOperationsStudio un TopologyImageRenderer qui transforme le modèle de topologie en PNG sans toucher à un seul contrôle Wisej. Donnez-lui une méthode Render qui prend la scène et une taille de sortie, crée le Bitmap, obtient son Graphics avec Graphics.FromImage, dessine les nœuds, les arêtes, les libellés et une légende avec des pens, des brushes et un GraphicsPath, et libère tout ce qu'il possède sans toucher aux objets de l'appelant. Résolvez sa Font via une méthode ResolveFont qui essaie une famille fournie par l'application et retombe sur une police connue comme présente, au lieu de supposer qu'une police système existe ; enregistrez le résultat en PNG dans un MemoryStream et confiez les octets à la page pour le téléchargement. Vérifiez ensuite les références du projet pour confirmer que vous n'ajoutez pas System.Drawing.Common à côté de l'implémentation managée de Wisej, exécutez ou préparez le renderer sous Windows et dans un conteneur Linux, et rédigez le comportement du même code sur les cibles iOS, Android et macOS Hybrid.

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

- ✓ TopologyImageRenderer sans aucune dépendance à un contrôle Wisej, qui rend via Graphics.FromImage dans un Bitmap
- ✓ Méthode ResolveFont avec une chaîne de repli explicite au lieu d'une police système supposée présente
- ✓ PNG écrit dans un MemoryStream et téléchargé depuis la page, chaque objet graphique possédé étant libéré
- ✓ Vérification des références confirmant que Managed.System.Drawing est utilisé et que System.Drawing.Common n'est pas ajouté à côté
- ✓ CrossPlatform.md avec les preuves sous Windows et en conteneur Linux, et les implications pour iOS, Android et macOS

Pourquoi il est conçu ainsi

Tout ce que VisualOperationsStudio a dessiné jusqu'ici appartient à une session vivante: la jauge se redessine parce qu'un contrôle l'a demandé, et la topologie Canvas du Module 5 n'existe que sous forme de commandes que le navigateur a déjà exécutées. L'exploitation veut désormais cette même topologie dans un courriel d'incident nocturne et jointe à un ticket, c'est-à-dire une image sans aucun contrôle derrière elle. Dans le même temps, l'application migre vers un conteneur Linux: le code de dessin doit donc continuer à fonctionner sur une machine sans GDI+ et sans libgdiplus.

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
System.Drawing.Common	Le wrapper GDI+ de Microsoft, spécifique à Windows depuis .NET 6; le commutateur Unix a disparu dans .NET 7, ce n'est donc pas une option pour Linux, macOS, iOS ou Android.
Managed.System.Drawing	L'identifiant du package NuGet, inversé parce que la racine de packages System.* est réservée; Wisej.NET 4 le référence déjà pour vous.
System.Drawing.Managed.dll	L'assembly que ce package livre — l'implémentation managée elle-même, que l'on voit dans un journal de compilation plutôt que dans vos sources.
L'espace de noms System.Drawing	Inchangé: Bitmap, Graphics, Pen et Font gardent leurs noms, si bien que le code de dessin existant compile sans réécriture.
Moteur graphique managé	Une implémentation purement managée sur un moteur ImageSharp personnalisé et intégré; sur .NET moderne, elle évite GDI+, libgdiplus et les appels natifs.
Graphics.FromImage	Le point d'entrée hors écran: une surface de dessin au-dessus d'un Bitmap qui vous appartient, sans contrôle, sans événement Paint et sans session.
PrivateFontCollection	Charge les fichiers de police que l'application distribue, pour que la sortie dépende de ce que vous avez déployé et non de ce qui se trouve installé sur la machine.

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 6. Vous pouvez ouvrir vous-même [module-06.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 VisualOperationsStudio 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 6 – « System.Drawing.Managed sur Windows, Linux, macOS, iOS et Android » – du cours LearnWisej « Dessin et peinture 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/drawing-and-painting/specs/module-06.md>

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

1. Projet « VisualOperationsStudio », 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 : Ajoutez à VisualOperationsStudio un TopologyImageRenderer qui transforme le modèle de topologie en PNG sans toucher à un seul contrôle Wisej. Donnez-lui une méthode Render qui prend la scène et une taille de sortie, crée le Bitmap, obtient son Graphics avec Graphics.FromImage, dessine les nœuds, les arêtes, les libellés et une légende avec des pens, des brushes et un GraphicsPath, et libère tout ce qu'il possède sans toucher aux objets de l'appelant. Résolvez sa Font via une méthode ResolveFont qui essaie une famille fournie par l'application et retombe sur une police connue comme présente, au lieu de supposer qu'une police système existe ; enregistrez le résultat en PNG dans un MemoryStream et confiez les octets à la page pour le téléchargement. Vérifiez ensuite les références du projet pour confirmer que vous n'ajoutez pas System.Drawing.Common à côté de l'implémentation managée de Wisej, exécutez ou préparez le renderer sous Windows et dans un conteneur Linux, et rédigez le comportement du même code sur les cibles iOS, Android et macOS Hybrid.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - TopologyImageRenderer sans aucune dépendance à un contrôle Wisej, qui rend via Graphics.FromImage dans un Bitmap
 - Méthode ResolveFont avec une chaîne de repli explicite au lieu d'une police système supposée présente
 - PNG écrit dans un MemoryStream et téléchargé depuis la page, chaque objet graphique possédé étant libéré
 - Vérification des références confirmant que Managed.System.Drawing est utilisé et que System.Drawing.Common n'est pas ajouté à côté
 - CrossPlatform.md avec les preuves sous Windows et en conteneur Linux, et les implications pour iOS, Android et macOS
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 collègue affirme que le `renderer` prouve que l'application est multiplateforme puisque le code compile sous Linux. Que devriez-vous réellement observer dans le conteneur avant d'accepter cette affirmation, et quelle partie de la sortie risque le plus de différer?
- Votre `renderer` libère le `Bitmap` qu'il a créé. Pourquoi libérer le `Graphics` dans le gestionnaire `Paint` d'un contrôle serait-il un bug, et quelle règle couvre les deux cas en une seule phrase?
- Quelqu'un propose de remplacer ce `renderer` par `SkiaSharp` parce qu'il obtient de meilleurs résultats en banc d'essai. Que devriez-vous savoir de la charge de travail avant d'accepter, et que perd l'équipe en abandonnant l'API `System.Drawing`?

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 collègue affirme que le `renderer` prouve que l'application est multiplateforme puisque le code compile sous Linux. Que devriez-vous réellement observer dans le conteneur avant d'accepter cette affirmation, et quelle partie de la sortie risque le plus de différer?
- ✓ Votre `renderer` libère le `Bitmap` qu'il a créé. Pourquoi libérer le `Graphics` dans le gestionnaire `Paint` d'un contrôle serait-il un bug, et quelle règle couvre les deux cas en une seule phrase?
- ✓ Quelqu'un propose de remplacer ce `renderer` par `SkiaSharp` parce qu'il obtient de meilleurs résultats en banc d'essai. Que devriez-vous savoir de la charge de travail avant d'accepter, et que perd l'équipe en abandonnant l'API `System.Drawing`?

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 VisualOperationsStudio 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/drawing-and-painting/specs/module-06.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 VisualOperationsStudio 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.