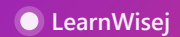


## Exercice de code avec l'IA

Construisez l'exemple AccessOps 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 l'audit et la gestion des secrets à AccessOps : un AuditService qui enregistre les approbations, les modifications de permissions et les téléchargements de documents avec l'utilisateur, l'heure, l'adresse et l'identifiant de corrélation, une table d'audit en ajout seul, un identifiant de corrélation créé pour chaque commande et écrit sur chaque ligne de log, les chaînes de connexion et le secret client OIDC déplacés dans la configuration d'environnement avec une procédure de rotation documentée, et une revue des logs qui masque les champs sensibles.

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

- ✓ AuditService et table d'audit en ajout seul
- ✓ Identifiant de corrélation dans chaque commande et chaque ligne de log
- ✓ Secrets déplacés dans la configuration d'environnement
- ✓ Procédure de rotation des secrets
- ✓ Revue du masquage des logs

### Pourquoi il est conçu ainsi

AccessOps authentifie désormais les personnes, résout leurs permissions et protège ses documents — et rien de tout cela n'aide le matin où quelqu'un demande qui a approuvé la demande 4471, depuis quelle adresse, et si quelqu'un a été refusé avant. Les cinq modules précédents ont décidé de ce qui est autorisé; celui-ci décide de ce que l'on pourra reconstituer après coup. Il comble aussi la brèche qui transforme un petit incident en incident majeur: un mot de passe de base de données ou un secret client OIDC dans un fichier commité il y a deux ans.

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                                                                                                                                                                                                  |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| piste d'audit              | Un enregistrement métier de chaque commande sensible — qui a agi, sur quoi, quand et d'où — conservé pendant une durée de rétention définie, parce que quelqu'un devra en répondre.                                    |
| stockage en ajout seul     | Une table d'audit dans laquelle le compte de l'application ne peut faire que des INSERT. Si le code qui écrit l'historique peut aussi le réécrire, l'historique ne prouve rien.                                        |
| preuve d'altération        | Une séquence sans trou associée à une chaîne de hachage, où le hachage de chaque ligne couvre ses propres champs et le hachage précédent, de sorte qu'une ligne modifiée rompt la chaîne.                              |
| identifiant de corrélation | Un identifiant créé au début d'une commande et répété sur chaque ligne de log et chaque ligne d'audit qu'elle provoque, pour qu'un simple clic puisse être reconstitué plus tard.                                      |
| scope de journalisation    | Un bloc de propriétés ambiant — <code>ILogger.BeginScope</code> ou un enrichir <code>Serilog</code> — qui attache l'identifiant de corrélation et l'acteur à tout ce qui est journalisé à l'intérieur.                 |
| stockage de secrets        | L'endroit où un identifiant vit en dehors du build: <code>user secrets</code> en local, variables d'environnement dans un conteneur, coffre managé en production — ou une identité managée, sans rien stocker du tout. |
| masquage                   | Une règle sur ce qui n'atteint jamais une destination de log — mots de passe, jetons, cookies, chaînes de connexion et données personnelles — appliquée par la journalisation structurée, pas par la mémoire.          |

## 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

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 AccessOps 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 - « Audit, secrets et configuration » - du cours LearnWisej « Modèles de sécurité et d'authentification », 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/security-and-auth-patterns/specs/module-06.md>

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

1. Projet « AccessOps », 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 l'audit et la gestion des secrets à AccessOps : un `AuditService` qui enregistre les approbations, les modifications de permissions et les téléchargements de documents avec l'utilisateur, l'heure, l'adresse et l'identifiant de corrélation, une table d'audit en ajout seul, un identifiant de corrélation créé pour chaque commande et écrit sur chaque ligne de log, les chaînes de connexion et le secret client OIDC déplacés dans la configuration d'environnement avec une procédure de rotation documentée, et une revue des logs qui masque les champs sensibles.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
  - `AuditService` et table d'audit en ajout seul
  - Identifiant de corrélation dans chaque commande et chaque ligne de log
  - Secrets déplacés dans la configuration d'environnement
  - Procédure de rotation des secrets
  - Revue du masquage des logs
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 :

- Votre table d'audit est en insertion seule pour l'application, et chaque ligne est chaînée par hachage. Nommez une action qu'un attaquant disposant du propre compte de base de données de l'application peut encore mener contre la piste, et expliquez précisément comment la vérification la ferait apparaître.
- Une approbation commence sur l'écran d'approbation et se termine dans une tâche en arrière-plan lancée avec `Application.StartTask` après la déconnexion de l'utilisateur. D'où cette tâche tire-t-elle son identifiant de corrélation et son acteur, et qu'est-ce qui casse si elle les lit dans la session?
- Un développeur retire le secret client OIDC du fichier de configuration suivi et pousse la

modification. Pourquoi le secret est-il toujours compromis, et que faut-il faire pour que l'incident soit réellement clos?

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 :

- ✓ Votre table d'audit est en insertion seule pour l'application, et chaque ligne est chaînée par hachage. Nommez une action qu'un attaquant disposant du propre compte de base de données de l'application peut encore mener contre la piste, et expliquez précisément comment la vérification la ferait apparaître.
- ✓ Une approbation commence sur l'écran d'approbation et se termine dans une tâche en arrière-plan lancée avec `Application.StartTask` après la déconnexion de l'utilisateur. D'où cette tâche tire-t-elle son identifiant de corrélation et son acteur, et qu'est-ce qui casse si elle les lit dans la session?
- ✓ Un développeur retire le secret client OIDC du fichier de configuration suivi et pousse la modification. Pourquoi le secret est-il toujours compromis, et que faut-il faire pour que l'incident soit réellement clos?

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/security-and-auth-patterns/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 AccessOps 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 AccessOps 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."*



### **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.