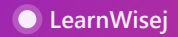


Exercice de code avec l'IA

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.



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

Mettez en place l'authentification d'AccessOps : une page de connexion avec verrouillage après cinq échecs, une vérification des mots de passe avec un hachage moderne salé, un SessionContext créé uniquement après la vérification et stocké dans la session, un délai d'inactivité qui renvoie l'utilisateur à la connexion et efface l'état, une déconnexion explicite qui met fin à la session, et un enregistrement d'audit pour chaque tentative de connexion.

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

- ✓ Page de connexion avec verrouillage et messages d'erreur génériques
- ✓ Service de hachage et de vérification des mots de passe
- ✓ SessionContext avec identité vérifiée
- ✓ Délai d'inactivité et déconnexion explicite
- ✓ Enregistrements d'audit des connexions

Pourquoi il est conçu ainsi

AccessOps contient des dossiers d'employés, des documents confidentiels et des demandes d'approbation: le premier écran que l'on atteint est donc aussi le premier contrôle qui compte. Tout ce que le reste de ce cours fait respecter — permissions, règles au niveau des enregistrements, pistes d'audit — repose sur une seule hypothèse: le serveur sait qui se trouve à l'autre bout de cette session et peut le prouver. Le Module 1 a tracé les frontières de confiance; ce module place une identité vérifiée à l'intérieur de la frontière de session et l'empêche d'être devinée, empruntée ou prolongée au-delà de sa durée de vie.

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
hachage lent et salé	Une dérivation à sens unique du mot de passe, avec un sel aléatoire propre à chaque utilisateur et un nombre d'itérations volontairement coûteux, de sorte qu'une table volée ne peut pas être inversée et que chaque tentative coûte un vrai travail.
comparaison à temps constant	Comparer deux hachages en un nombre fixe d'opérations avec <code>CryptographicOperations.FixedTimeEquals</code> , afin que le temps de réponse du serveur ne révèle jamais combien d'octets d'une tentative étaient justes.
SessionContext	L'objet créé uniquement après la vérification d'un mot de passe, qui contient l'id de l'employé, son département, ses rôles et les détails de sa connexion, et qui est stocké dans <code>Application.Session</code> comme l'unique endroit où le code demande qui est l'utilisateur.
expiration de session	La période d'inactivité au terme de laquelle Wisej.NET libère la session côté serveur et tout ce qu'elle contient, configurée pour l'application dans <code>Default.json</code> plutôt que devinée écran par écran.
fixation de session	Une attaque dans laquelle l'identifiant qui portait la page de connexion anonyme porte ensuite aussi la session authentifiée, si bien qu'un identifiant planté par l'attaquant devient un identifiant connecté.
limitation des tentatives de connexion	Compter les échecs par compte et par adresse client et refuser les tentatives suivantes pendant une période de verrouillage: c'est ce qui transforme une machine à deviner illimitée en quelques essais par heure.
enregistrement d'audit de connexion	Une ligne par tentative avec le nom d'utilisateur essayé, l'heure, l'adresse client et le résultat réel — l'endroit où la raison précise de l'échec est consignée, puisque l'écran n'en affiche jamais qu'une version générique.

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 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 2 - « Authentification et identité de session » - 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-02.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 : Mettez en place l'authentification d'AccessOps : une page de connexion avec verrouillage après cinq échecs, une vérification des mots de passe avec un hachage moderne salé, un `SessionContext` créé uniquement après la vérification et stocké dans la session, un délai d'inactivité qui renvoie l'utilisateur à la connexion et efface l'état, une déconnexion explicite qui met fin à la session, et un enregistrement d'audit pour chaque tentative de connexion.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Page de connexion avec verrouillage et messages d'erreur génériques
 - Service de hachage et de vérification des mots de passe
 - `SessionContext` avec identité vérifiée
 - Délai d'inactivité et déconnexion explicite
 - Enregistrements d'audit des connexions
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 page de connexion répond à un mauvais mot de passe en 480 ms et à un nom d'utilisateur inconnu en 6 ms. Expliquez ce qu'un attaquant apprend de cette seule différence, et décrivez les deux modifications qui suppriment ce signal sans changer le message.
- Un collègue propose de conserver l'employé connecté dans une propriété statique, parce que « c'est plus simple que d'aller chercher dans la session à chaque fois ». Décrivez précisément ce que voit le deuxième utilisateur simultané, et à quel moment les deux sessions interfèrent pour la première fois.
- Votre bouton Logout efface le `SessionContext` mais ne met pas fin à la session. Nommez deux éléments encore récupérables ensuite, et dites lequel vous jugez le plus grave dans un

portail qui affiche des documents confidentiels.

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 page de connexion répond à un mauvais mot de passe en 480 ms et à un nom d'utilisateur inconnu en 6 ms. Expliquez ce qu'un attaquant apprend de cette seule différence, et décrivez les deux modifications qui suppriment ce signal sans changer le message.
- ✓ Un collègue propose de conserver l'employé connecté dans une propriété statique, parce que « c'est plus simple que d'aller chercher dans la session à chaque fois ». Décrivez précisément ce que voit le deuxième utilisateur simultané, et à quel moment les deux sessions interfèrent pour la première fois.
- ✓ Votre bouton Logout efface le SessionContext mais ne met pas fin à la session. Nommez deux éléments encore récupérables ensuite, et dites lequel vous jugez le plus grave dans un portail qui affiche des documents confidentiels.

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.

- ✓ **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 gérait pas."

- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet AccessOps 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."*



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.