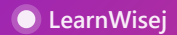


Exercice de code avec l'IA

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.



Écrire le code, c'est désormais la moitié facile.

Obtenir d'un assistant moderne du code qui fonctionne pour « Intégration d'OAuth et d'OpenID Connect » ne demande qu'un bon prompt. Le juger, le corriger et l'étendre demande tout ce que ce module a couvert. C'est sur ce partage que repose l'exercice : vous apportez la spécification et la relecture, l'assistant apporte la frappe.

Ce que vous demandez

Ajoutez la connexion externe à AccessOps : un client OpenID Connect qui utilise le flux authorization code avec PKCE face à un fournisseur d'identité de test, un handler de callback qui valide l'ID token et son nonce, un mappage des claims qui remplit le SessionContext et crée un enregistrement d'employé local à la première connexion, un repli sur la connexion locale pour les comptes de service, et une déconnexion qui met fin à la fois à la session locale et à celle du fournisseur.

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

- ✓ Configuration du client OIDC avec PKCE
- ✓ Gestion du callback avec validation du jeton et du nonce
- ✓ Mappage des claims et provisionnement à la première connexion
- ✓ Repli sur la connexion locale pour les comptes de service
- ✓ Déconnexion couvrant les deux sessions

Pourquoi il est conçu ainsi

AccessOps sait déjà vérifier ses propres mots de passe, mais l'organisation qui l'exploite ne veut pas d'un second annuaire: le personnel se connecte à tout le reste avec le fournisseur d'identité de l'entreprise, et les arrivées, les départs et la politique d'authentification multifacteur sont administrés en un seul endroit. Déléguer la connexion supprime entièrement le mot de passe d'AccessOps et le remplace par une autre obligation: le portail reçoit désormais une identité d'un tiers, par l'intermédiaire du navigateur de l'utilisateur, et doit prouver que cette identité est authentique, non modifiée et bien destinée à lui avant d'afficher le moindre enregistrement de département.

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
flux authorization code	Le flux OpenID Connect dans lequel le navigateur ne transporte jamais qu'un code à usage unique et de courte durée, tandis que les jetons sont récupérés par votre serveur via un appel HTTPS direct que le navigateur ne voit jamais.
PKCE	Un code verifier aléatoire généré à chaque connexion, envoyé au point de terminaison autorize uniquement sous forme de hachage SHA-256 et révélé au point de terminaison token, de sorte qu'un code intercepté ne peut être échangé par personne d'autre.
ID token	Un JWT signé qui atteste que ce fournisseur a authentifié ce sujet, pour ce client, à ce moment. C'est le seul jeton qu'AccessOps consulte pour décider qui est l'utilisateur.
access token et refresh token	L'access token est un identifiant au porteur permettant d'appeler l'API de quelqu'un d'autre; le refresh token permet d'obtenir des remplaçants via le canal arrière. Ni l'un ni l'autre ne prouve l'identité pour votre propre application.
state et nonce	Deux valeurs aléatoires que vous générez à chaque connexion: state rattache le callback à la requête que vous avez réellement lancée, nonce rattache l'ID token renvoyé à cette même requête, pour qu'un jeton plus ancien ne puisse pas y être rejoué.
redirect URI	L'URL exacte, préalablement enregistrée, vers laquelle le fournisseur renvoie le navigateur. Dans AccessOps, il s'agit d'un petit endpoint côté serveur plutôt que d'une fenêtre Wisej.NET, car il doit s'exécuter avant qu'aucune UI n'existe.
mappage des claims et provisionnement	Transformer les claims du fournisseur en employé AccessOps: le couple émetteur / sujet identifie le compte, et une ligne locale est créée à la première connexion, sans aucune permission tant que quelqu'un ne les accorde pas.

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 4. Vous pouvez ouvrir vous-même [module-04.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 4 - « Intégration d'OAuth et d'OpenID Connect » - 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-04.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 la connexion externe à AccessOps : un client OpenID Connect qui utilise le flux authorization code avec PKCE face à un fournisseur d'identité de test, un handler de callback qui valide l'ID token et son nonce, un mappage des claims qui remplit le `SessionContext` et crée un enregistrement d'employé local à la première connexion, un repli sur la connexion locale pour les comptes de service, et une déconnexion qui met fin à la fois à la session locale et à celle du fournisseur.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Configuration du client OIDC avec PKCE
 - Gestion du callback avec validation du jeton et du nonce
 - Mappage des claims et provisionnement à la première connexion
 - Repli sur la connexion locale pour les comptes de service
 - Déconnexion couvrant les deux sessions
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 attaquant récupère le code d'autorisation dans l'historique d'un navigateur sur un poste partagé. Qu'est-ce qui l'empêche de l'échanger contre des jetons, et qu'est-ce qui n'aurait pas suffi à l'en empêcher sans PKCE?
- Votre fournisseur émet des claims de groupe qui ressemblent exactement aux noms des permissions d'AccessOps. Pourquoi le `PermissionService` résout-il malgré tout les permissions lui-même au lieu de copier ces claims directement dans le `SessionContext`?
- Un utilisateur clique sur Sign out, ne ferme rien, puis clique immédiatement sur Sign in - et se retrouve dans AccessOps sans rien saisir. Quelle session a survécu, et quelle redirection manquait?

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 attaquant récupère le code d'autorisation dans l'historique d'un navigateur sur un poste partagé. Qu'est-ce qui l'empêche de l'échanger contre des jetons, et qu'est-ce qui n'aurait pas suffi à l'en empêcher sans PKCE?
- ✓ Votre fournisseur émet des claims de groupe qui ressemblent exactement aux noms des permissions d'AccessOps. Pourquoi le PermissionService résout-il malgré tout les permissions lui-même au lieu de copier ces claims directement dans le SessionContext?
- ✓ Un utilisateur clique sur Sign out, ne ferme rien, puis clique immédiatement sur Sign in — et se retrouve dans AccessOps sans rien saisir. Quelle session a survécu, et quelle redirection manquait?

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 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."*
- ✓ **Ajoutez le deuxième cas.** *"Étends ce projet AccessOps 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/security-and-auth-patterns/specs/module-04.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.