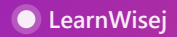


Exercice de code avec l'IA

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.



Briefez-le, lisez-le, puis cassez-le.

Cet exercice comporte trois parties, et seule la première concerne le prompt. Vous briefez un assistant avec la spécification du module et récupérez une solution AccessOps ; vous lisez ce qu'il a écrit et décidez s'il fait vraiment ce que le module demandait ; puis vous le poussez plus loin que ce que le module demandait. C'est dans la deuxième et la troisième partie que l'on apprend.

Ce que vous demandez

Ajoutez l'autorisation à AccessOps : un `PermissionService` qui résout les rôles et les claims en permissions à la connexion, une garde `Authorize` appelée par chaque méthode de `DocumentService` et d'`ApprovalService`, une règle au niveau des enregistrements selon laquelle les employés ne voient que leur département et les managers que leurs collaborateurs directs, une UI qui masque les actions d'administration et désactive les boutons d'approbation que l'utilisateur ne peut pas utiliser, et un test prouvant qu'une requête falsifiée émise par un non-manager est rejetée par le service.

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

- ✓ `PermissionService` résolvant les permissions dans le `SessionContext`
- ✓ Garde d'autorisation dans chaque méthode de service
- ✓ Règles au niveau des enregistrements par département et par manager
- ✓ Adaptation de l'UI selon les permissions
- ✓ Test négatif avec une requête falsifiée

Pourquoi il est conçu ainsi

AccessOps sait désormais qui est connecté: le Module 2 a laissé une identité vérifiée dans le `SessionContext`. Savoir qui est quelqu'un ne dit rien de ce qu'il peut faire d'un document confidentiel ou d'une demande d'approbation à six chiffres. Dans un portail interne, l'incident type est rarement une connexion défaillante — c'est un gestionnaire de paie qui peut lire les contrats du service juridique parce que la requête n'a jamais été restreinte, ou un chef d'équipe qui approuve sa propre note de frais parce que le seul contrôle était un bouton désactivé.

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
rôle	Une fonction métier nommée, à gros grain, comme Employee, Manager ou Auditor. Les rôles sont pratiques à administrer mais trop grossiers pour être testés directement, car un écran a besoin de connaître une action, pas un métier entier.
claim	Un fait nom/valeur sur l'identité vérifiée — département 14, centre de coût EU-North, manager de l'équipe 7. Les claims portent les données dont les règles au niveau des enregistrements ont besoin, et ils proviennent du référentiel d'identités, jamais du navigateur.
permission	Une action nommée que l'application peut effectuer, comme ApproveRequest ou ViewDepartmentDocuments. Le code teste des permissions; les administrateurs attribuent des rôles; la correspondance entre les deux relève de la configuration.
résolution des permissions	L'étape unique, à la connexion, qui développe les rôles et les claims en un ensemble de permissions plat et immuable conservé dans le SessionContext, afin qu'aucune requête ne relise la table des rôles et que deux contrôles ne puissent jamais se contredire.
garde d'autorisation	Un appel tel que Require(Permission.ApproveRequest) en tête d'une méthode de service, qui lève une exception avant que le moindre travail ne commence. C'est le seul endroit où l'application de la règle a réellement lieu.
règle au niveau des enregistrements	Une restriction sur les lignes qu'une action autorisée peut toucher — votre propre département, vos propres collaborateurs directs — appliquée dans la requête ou la commande plutôt qu'après le chargement des données.
adaptation de l'UI	Masquer, désactiver ou expliquer une action à partir du même ensemble de permissions que celui appliqué par le serveur. C'est une attention envers l'utilisateur, jamais un contrôle.

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 3. Vous pouvez ouvrir vous-même [module-03.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 3 - « Autorisation : rôles, claims et permissions » - 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-03.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'autorisation à AccessOps : un `PermissionService` qui résout les rôles et les claims en permissions à la connexion, une garde `Authorize` appelée par chaque méthode de `DocumentService` et d'`ApprovalService`, une règle au niveau des enregistrements selon laquelle les employés ne voient que leur département et les managers que leurs collaborateurs directs, une UI qui masque les actions d'administration et désactive les boutons d'approbation que l'utilisateur ne peut pas utiliser, et un test prouvant qu'une requête falsifiée émise par un non-manager est rejetée par le service.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - `PermissionService` résolvant les permissions dans le `SessionContext`
 - Garde d'autorisation dans chaque méthode de service
 - Règles au niveau des enregistrements par département et par manager
 - Adaptation de l'UI selon les permissions
 - Test négatif avec une requête falsifiée
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 auditeur reçoit des droits d'approbation alors que sa session `AccessOps` est encore ouverte. Expliquez précisément à quel moment la nouvelle permission prend effet avec votre conception, pourquoi vous avez choisi ce moment, et ce qu'il faudrait modifier pour qu'elle soit immédiate.
- Votre requête de documents filtre par département dans le repository. Décrivez ce qu'un attaquant gagne si ce filtre est déplacé vers le `DataSource` de la grille, alors même que le résultat visible à l'écran est identique.
- `ApprovalService` comme son repository pourraient vérifier que l'approbateur est le manager

du demandeur. Défendez l'un des deux comme le bon emplacement, et dites ce qui casse si la vérification se trouve dans l'autre, ou dans les deux.

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 auditeur reçoit des droits d'approbation alors que sa session AccessOps est encore ouverte. Expliquez précisément à quel moment la nouvelle permission prend effet avec votre conception, pourquoi vous avez choisi ce moment, et ce qu'il faudrait modifier pour qu'elle soit immédiate.
- ✓ Votre requête de documents filtre par département dans le repository. Décrivez ce qu'un attaquant gagne si ce filtre est déplacé vers le DataSource de la grille, alors même que le résultat visible à l'écran est identique.
- ✓ ApprovalService comme son repository pourraient vérifier que l'approbateur est le manager du demandeur. Défendez l'un des deux comme le bon emplacement, et dites ce qui casse si la vérification se trouve dans l'autre, ou dans les deux.

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.

- ✓ **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."*
- ✓ **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."*



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.