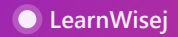


Ejercicio de programación con IA

Construye el ejemplo AccessOps de este módulo con ChatGPT o Claude a partir de un prompt listo para usar que dirige al modelo a la especificación del módulo, y después revisa, ejecuta y amplía lo que te devuelva.



No vas a teclear este ejemplo. Vas a dirigirlo.

Ya casi nadie escribe código Wisej.NET partiendo de un archivo vacío — un asistente escribe el primer borrador y el desarrollador decide si es correcto. Eso solo funciona si dominas el tema, y eso es lo que te dio el resto de este módulo. Aquí le pasas ese conocimiento a ChatGPT o a Claude, recibes el ejemplo de este módulo como una solución de Visual Studio que funciona y lo revisas como revisarías el pull request de un compañero.

Qué estás pidiendo

Añade auditoría y gestión de secretos a AccessOps: un AuditService que registra aprobaciones, cambios de permisos y descargas de documentos con el usuario, la hora, la dirección y el ID de correlación; una tabla de auditoría de solo anexado; un ID de correlación creado por comando y escrito en cada línea de log; las cadenas de conexión y el secreto de cliente OIDC trasladados a la configuración del entorno con un procedimiento de rotación documentado; y una revisión de los logs que enmascara los campos sensibles.

El módulo pide estos entregables, y el prompt de abajo le pide al asistente exactamente esos:

- ✓ AuditService y tabla de auditoría de solo anexado
- ✓ ID de correlación en cada comando y cada línea de log
- ✓ Secretos trasladados a la configuración del entorno
- ✓ Procedimiento de rotación de secretos
- ✓ Revisión del enmascaramiento de los logs

Por qué tiene esta forma

AccessOps ya autentica a las personas, resuelve sus permisos y protege sus documentos, y nada de eso sirve la mañana en que alguien pregunta quién aprobó la solicitud 4471, desde qué dirección y si antes se rechazó a alguien. Los cinco módulos anteriores decidieron lo que está permitido; este decide lo que se puede reconstruir después. También cierra el hueco que convierte un incidente pequeño en uno grande: una contraseña de base de datos o un secreto de cliente OIDC en un archivo confirmado en el repositorio hace dos años.

Estos son los términos que el asistente tiene que dominar. Si lo que te devuelve los usa a la ligera, no ha entendido el módulo:

Término	Qué significa aquí
registro de auditoría	Un registro de negocio de cada comando sensible —quién actuó, sobre qué, cuándo y desde dónde— conservado durante un periodo de retención establecido, porque alguien tendrá que responder de ello.
almacenamiento de solo anexo	Una tabla de auditoría en la que la cuenta de la aplicación solo puede hacer INSERT. Si el código que escribe el historial también puede reescribirlo, el historial no demuestra nada.
evidencia de manipulación	Una secuencia sin huecos más una cadena de hashes, en la que el hash de cada fila cubre sus propios campos y el hash anterior, de modo que una fila editada rompe la cadena.
ID de correlación	Un identificador generado cuando empieza un comando y repetido en cada línea de log y cada fila de auditoría que provoca, para poder reconstruir después un único clic.
ámbito de logging	Un bloque de propiedades ambiental — ILogger.BeginScope o un enricher de Serilog— que añade el ID de correlación y el actor a todo lo que se registra dentro de él.
almacén de secretos	El lugar donde vive una credencial fuera del build: user secrets en local, variables de entorno en un contenedor, un almacén gestionado en producción, o una identidad administrada, sin nada almacenado en absoluto.
enmascaramiento	Una regla sobre lo que nunca llega a un destino de logs —contraseñas, tokens, cookies, cadenas de conexión y datos personales— aplicada mediante logging estructurado, no mediante la memoria.

Constrúyelo — elige tu asistente

Los dos botones abren una pestaña nueva con el **mismo prompt listo para usar** ya escrito. El prompt dirige al modelo a la especificación de este módulo — un único archivo Markdown con la lección, el enunciado del laboratorio y los entregables — para que trabaje con el material real en lugar de adivinar de qué trata el Módulo 6. Puedes abrir tú mismo [module-06.md](#) para ver exactamente lo que leerá.



Usar ChatGPT

Abre [chatgpt.com](#) con el prompt de este módulo ya cargado. Funciona con GPT-5 o Codex; cualquier plan que pueda navegar por la web descargará la especificación.



Usar Claude

Abre [claude.ai](#) con el mismo prompt ya cargado. Si usas Claude Code, puedes pegarlo en una terminal abierta en una carpeta vacía.



Envíalo tal cual y obtendrás un proyecto, no un fragmento

El prompt termina pidiendo toda la solución como un único `.zip` con los pasos para compilarla y ejecutarla. Descárgalo, abre la solución en Visual Studio, pulsa F5 y AccessOps se abrirá en `localhost` . Necesitas el SDK de .NET y el paquete NuGet `Wisej-4` ; todo lo demás, el prompt le pide al asistente que lo escriba él mismo.

Estoy siguiendo el Módulo 6 - «Auditoría, secretos y configuración» - del curso de LearnWisej «Patrones de seguridad y autenticación», y quiero construir contigo el ejemplo de este módulo. Respóndeme en español.

Empieza leyendo la especificación del módulo. Es un único archivo Markdown (en inglés) que contiene la lección completa, el enunciado del laboratorio y los entregables:

<https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-06.md>

Después construye un proyecto Wisej.NET 4 completo que pueda abrir en Visual Studio:

1. Proyecto «AccessOps», C#, net8.0 o posterior, paquete NuGet Wisej-4 4.1.x, host Kestrel que llame a `app.UseWisej()`.
2. Construye lo que pide el laboratorio de la especificación: Añade auditoría y gestión de secretos a AccessOps: un `AuditService` que registra aprobaciones, cambios de permisos y descargas de documentos con el usuario, la hora, la dirección y el ID de correlación; una tabla de auditoría de solo anexo; un ID de correlación creado por comando y escrito en cada línea de log; las cadenas de conexión y el secreto de cliente OIDC trasladados a la configuración del entorno con un procedimiento de rotación documentado; y una revisión de los logs que enmascara los campos sensibles.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `AuditService` y tabla de auditoría de solo anexo
 - ID de correlación en cada comando y cada línea de log
 - Secretos trasladados a la configuración del entorno
 - Procedimiento de rotación de secretos
 - Revisión del enmascaramiento de los logs
4. Sigue la especificación siempre que nombre una clase, propiedad o evento de Wisej.NET - usa exactamente ese. Donde deje la elección abierta, dime qué API elegiste y por qué.
5. Mantén la autoridad en el servidor: el estado y la validación viven en .NET, y el navegador muestra e informa. Nunca traslades una decisión a JavaScript porque allí resulte más fácil.
6. Haz que el resultado se pueda demostrar: debo poder ejecutarlo y ver el comportamiento del que trata este módulo, incluido lo que ocurre cuando algo sale mal.

Antes de escribir una sola línea de código, explica con tus propias palabras qué vas a construir y dónde vivirá cada pieza, para que pueda corregir el diseño mientras todavía es barato.

Cuando el código esté terminado, empaqueta toda la solución en un único archivo .zip descargable y dame los pasos exactos para compilarla y ejecutarla.

Por último, responde a estas preguntas de la especificación usando tu propia implementación como prueba:

- Tu tabla de auditoría es de solo inserción para la aplicación y cada fila está encadenada por hash. Nombra algo que un atacante con el propio login de base de datos de la aplicación aún puede hacerle al registro y explica exactamente cómo lo pondría de manifiesto el proceso de verificación.
- Una aprobación se inicia en la pantalla de aprobaciones y termina en una tarea en segundo plano iniciada con `Application.StartTask` después de que el usuario haya cerrado sesión. ¿De dónde obtiene esa tarea su ID de correlación y su actor, y qué se rompe si los lee de la sesión?
- Un desarrollador elimina el secreto de cliente OIDC del archivo de configuración versionado y sube el cambio. ¿Por qué el secreto sigue comprometido y qué tiene que ocurrir

antes de que el incidente quede realmente cerrado?

Pregúntame por cualquier cosa ambigua antes de empezar.

Después revisa lo que te devuelve

Que el código compile no significa que sea correcto. Las preguntas interesantes son de diseño: ¿es este el control de Wisej.NET que la tarea realmente necesita?, ¿sigue el ciclo de vida que te ofrece el framework?, ¿el estado de referencia está en el servidor o se quedó en el navegador porque era más cómodo? Lee el resultado como revisor, no como cliente.

Este módulo te da la lista de comprobación. Responde a cada punto basándote en el código que acabas de recibir:

- ✓ Tu tabla de auditoría es de solo inserción para la aplicación y cada fila está encadenada por hash. Nombra algo que un atacante con el propio login de base de datos de la aplicación aún puede hacerle al registro y explica exactamente cómo lo pondría de manifiesto el proceso de verificación.
- ✓ Una aprobación se inicia en la pantalla de aprobaciones y termina en una tarea en segundo plano iniciada con `Application.StartTask` después de que el usuario haya cerrado sesión. ¿De dónde obtiene esa tarea su ID de correlación y su actor, y qué se rompe si los lee de la sesión?
- ✓ Un desarrollador elimina el secreto de cliente OIDC del archivo de configuración versionado y sube el cambio. ¿Por qué el secreto sigue comprometido y qué tiene que ocurrir antes de que el incidente quede realmente cerrado?

Todo lo que hayas tenido que corregir merece quedar por escrito. Una nota breve con las convenciones y decisiones que acordaste, pegada en el primer mensaje de tu próxima sesión, vale más que cualquier prompt ingenioso.



Todavía no hay un ejemplo de referencia para este curso — el tuyo es el primero

El [repositorio de ejemplos de LearnWisej](#) tiene soluciones completas para la mayoría de los cursos, pero todavía no para este. Eso significa que no hay nada con qué compararte ni ningún atajo: la especificación y tu propia revisión son toda tu red de seguridad. Guarda lo que construyas — y si sale bien, nos encantaría verlo.

Tres prompts para ir más allá

Dirige tu asistente al proyecto que acabas de construir y prueba estos. Cada uno es un ejercicio de revisión y ampliación, no una reescritura.

- ✓ **Demuestra que de verdad funciona.** *"Recorre este proyecto entregable por entregable frente a la especificación del módulo en <https://learnwisej.com/assets/courses/security-and-auth-patterns/specs/module-06.md> y muéstrame la prueba de que cada uno está realmente terminado — la ruta del código, dónde hacer clic y qué debería ver. Si un entregable solo se cumple en parte, dílo claramente en lugar de justificarlo. Después añade lo que falte."*
- ✓ **Revísalo como un pull request.** *"Explícame este código de AccessOps como si estuvieras revisando el pull request de un compañero y no lo hubieras escrito tú. ¿Dónde vive el estado?, ¿qué*

ocurre de principio a fin con una acción del usuario?, ¿qué devolverías para que lo cambien? y ¿qué partes no te sentirías cómodo poniendo delante de un cliente? Sé concreto y cita las líneas a las que te refieres — no me resumas el diseño."

- ✓ **Cambia una pieza sin mover las juntas.** *"Elige la parte de este proyecto AccessOps que sea más claramente un sustituto de algo real — una fuente de datos falsa, un almacén en memoria, un servicio simplificado — y reemplázala por una implementación con forma de producción. El contrato a ambos lados no debe cambiar: los mismos tipos, las mismas llamadas, el mismo comportamiento ante fallos. Dime qué te obligó a gestionar la implementación real que el sustituto no gestionaba."*



Llévalo al laboratorio

El laboratorio práctico que cierra este módulo pide los mismos entregables que acabas de hacer que generara el asistente. Usa lo que tienes aquí como forma de tu respuesta y como prueba que la respalda — no como la respuesta en sí. El laboratorio se califica por tu razonamiento, y esa parte tiene que ser tuya.