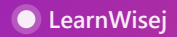


Ejercicio de programación con IA

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 devuelve.



Escribir el código ya es la mitad fácil.

Conseguir que un asistente moderno genere código que funcione para «Integración con OAuth y OpenID Connect» solo requiere un buen prompt. Juzgarlo, corregirlo y ampliarlo requiere todo lo que ha cubierto este módulo. Ese es el reparto sobre el que se construye este ejercicio: tú aportas la especificación y la revisión; el asistente, el tecleo.

Qué estás pidiendo

Añade el inicio de sesión externo a AccessOps: un cliente OpenID Connect que usa el flujo de código de autorización con PKCE contra un proveedor de identidades de prueba, un controlador de callback que valida el token de ID y su nonce, una asignación de claims que rellena el SessionContext y aprovisiona una ficha de empleado local en el primer inicio de sesión, una alternativa de inicio de sesión local para las cuentas de servicio y un cierre de sesión que termina tanto la sesión local como la del proveedor.

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

- ✓ Configuración del cliente OIDC con PKCE
- ✓ Gestión del callback con validación del token y del nonce
- ✓ Asignación de claims y aprovisionamiento en el primer inicio de sesión
- ✓ Inicio de sesión local alternativo para cuentas de servicio
- ✓ Cierre de sesión que cubre ambas sesiones

Por qué tiene esta forma

AccessOps ya puede verificar una contraseña propia, pero la organización que lo utiliza no quiere un segundo directorio: el personal inicia sesión en todo lo demás con el proveedor de identidades corporativo, y las altas, las bajas y la política de autenticación multifactor se administran en un único lugar. Delegar el inicio de sesión elimina por completo la contraseña de AccessOps y la sustituye por otra obligación: ahora un tercero entrega una identidad al portal, a través del propio navegador del usuario, y el portal tiene que demostrar que esa identidad es auténtica, no ha sido modificada y va dirigida a él antes de mostrar una sola ficha de departamento.

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
flujo de código de autorización	El flujo de OpenID Connect en el que el navegador solo lleva un código de un solo uso y vida corta, mientras que los tokens los obtiene tu servidor mediante una llamada HTTPS directa que el navegador nunca ve.
PKCE	Un code verifier aleatorio generado en cada inicio de sesión, enviado al endpoint de autorización solo como su hash SHA-256 y revelado en el endpoint de token, para que nadie más pueda canjear un código interceptado.
token de ID	Un JWT firmado que afirma que este proveedor autenticó a este sujeto, para este cliente, en este momento. Es el único token que AccessOps consulta para decidir quién es el usuario.
tokens de acceso y de actualización	El token de acceso es una credencial al portador para llamar a la API de otro; el token de actualización obtiene reemplazos por el canal trasero. Ninguno de los dos es prueba de identidad para tu propia aplicación.
state y nonce	Dos valores aleatorios que generas en cada inicio de sesión: state vincula el callback a la solicitud que realmente iniciaste, y nonce vincula el token de ID devuelto a esa misma solicitud para que no se pueda reinyectar en ella un token anterior.
URI de redirección	La URL exacta, registrada de antemano, a la que el proveedor devuelve el navegador. En AccessOps es un pequeño endpoint del lado del servidor y no una ventana de Wisej.NET, porque tiene que ejecutarse antes de que exista ninguna interfaz.
asignación de claims y aprovisionamiento	Convertir los claims del proveedor en un empleado de AccessOps: el par emisor y sujeto identifica la cuenta, y en el primer inicio de sesión se crea una fila local sin permisos hasta que alguien los conceda.

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 4. Puedes abrir tú mismo [module-04.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 4 - «Integración con OAuth y OpenID Connect» - 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-04.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 el inicio de sesión externo a AccessOps: un cliente OpenID Connect que usa el flujo de código de autorización con PKCE contra un proveedor de identidades de prueba, un controlador de callback que valida el token de ID y su nonce, una asignación de claims que rellena el `SessionContext` y aprovisiona una ficha de empleado local en el primer inicio de sesión, una alternativa de inicio de sesión local para las cuentas de servicio y un cierre de sesión que termina tanto la sesión local como la del proveedor.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Configuración del cliente OIDC con PKCE
 - Gestión del callback con validación del token y del nonce
 - Asignación de claims y aprovisionamiento en el primer inicio de sesión
 - Inicio de sesión local alternativo para cuentas de servicio
 - Cierre de sesión que cubre ambas sesiones
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:

- Un atacante obtiene el código de autorización del historial del navegador en un equipo compartido. ¿Qué le impide canjearlo por tokens, y qué no habría logrado detenerlo sin PKCE?
- Tu proveedor emite claims de grupo idénticos a los nombres de permiso de AccessOps. ¿Por qué el `PermissionService` sigue resolviendo los permisos por sí mismo en lugar de copiar esos claims directamente en el `SessionContext`?
- Un usuario pulsa Sign out, no cierra nada y vuelve a pulsar Sign in de inmediato, y aterrizas de nuevo dentro de AccessOps sin escribir nada. ¿Qué sesión ha sobrevivido y qué redirección faltaba?

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:

- ✓ Un atacante obtiene el código de autorización del historial del navegador en un equipo compartido. ¿Qué le impide canjearlo por tokens, y qué no habría logrado detenerlo sin PKCE?
- ✓ Tu proveedor emite claims de grupo idénticos a los nombres de permiso de AccessOps. ¿Por qué el PermissionService sigue resolviendo los permisos por sí mismo en lugar de copiar esos claims directamente en el SessionContext?
- ✓ Un usuario pulsa Sign out, no cierra nada y vuelve a pulsar Sign in de inmediato, y aterriza de nuevo dentro de AccessOps sin escribir nada. ¿Qué sesión ha sobrevivido y qué redirección faltaba?

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.

- ✓ **Haz que aguante un mal día.** *"Revisa este proyecto AccessOps en busca de los modos de fallo que todavía no gestiona — una entrada incorrecta o ausente, una dependencia lenta o no disponible, trabajo que llega después de que el usuario haya salido de la pantalla y dos acciones que compiten entre sí. Para cada uno, dime primero si es un riesgo real en Wisej.NET 4 y por qué, y solo después escribe la corrección. Si decides que algo no necesita gestión, dilo y explica el razonamiento en lugar de añadir código defensivo."*
- ✓ **Añade el segundo caso.** *"Amplía este proyecto AccessOps con un segundo caso que siga el mismo patrón que el primero — otra entidad, pantalla o control con la misma forma, conectado de la misma manera. Extrae a una clase base o a un servicio lo que de verdad sea común, pero solo*

donde se justifique; dos cosas parecidas no siempre piden una abstracción. Muéstrame el diff y justifica cada cosa que hayas extraído."

- ✓ **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-04.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, dilo claramente en lugar de justificarlo. Después añade lo que falte."*



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.