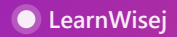


Ejercicio de programación con IA

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

Crea la solución AccessOps y redacta su modelo de amenazas: dibuja los límites de confianza, enumera los activos (fichas de empleados, documentos, aprobaciones), identifica los puntos de entrada (inicio de sesión, subidas, descargas, endpoints personalizados, el asistente de IA), valora cada amenaza y asigna cada mitigación al módulo de este curso que la implementa. Construye el esqueleto de la aplicación con una página de inicio de sesión provisional.

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

- ✓ Solución AccessOps con una página de inicio de sesión provisional
- ✓ Diagrama de límites de confianza
- ✓ Inventario de activos y puntos de entrada
- ✓ Tabla de amenazas con probabilidad, impacto y mitigación
- ✓ Mapa de ubicación de los controles de seguridad

Por qué tiene esta forma

AccessOps es un portal interno de operaciones: departamentos, empleados, documentos confidenciales, solicitudes de aprobación y una consola de administración. Todo el que puede llegar a él ya trabaja para la empresa, y precisamente por eso el trabajo de seguridad se aplaza, y por eso un único portal acaba conteniendo bandas salariales, cartas disciplinarias y el historial de aprobaciones de cada excepción que la empresa ha concedido. Antes de que exista una sola línea de código de autenticación necesitas tres respuestas: qué tiene el navegador, por dónde pasan los límites y cuáles de las cosas que podrían salir mal merecen protegerse primero.

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
modelo de interfaz del lado del servidor	Los controles son objetos C# en el proceso del servidor; el navegador tiene un reflejo renderizado y un canal de eventos, así que tu código y tus consultas nunca se descargan.
carga útil del estado del widget	Los valores de propiedades que Wisej.NET serializa para que el navegador pueda dibujar un control. Todo lo que asignas viaja por la red, también en los controles que el usuario no ve.
límite de confianza	Una línea donde los datos pasan entre partes con distinto nivel de confianza. Todo lo que la cruza lo vuelve a comprobar el lado más fiable, porque el otro lado puede mentir.
límite de sesión	La frontera entre un navegador y su propio estado en el servidor, en Application.Session. Las decisiones salen del estado que está en el lado del servidor, nunca de un valor del navegador.
punto de entrada	Cualquier vía de acceso que un atacante pueda alcanzar: la página de inicio de sesión, los controladores de eventos, las subidas, las descargas, los handlers HTTP personalizados, las llamadas de interoperabilidad y el prompt del asistente de IA.
STRIDE	Seis categorías para nombrar amenazas —suplantación, manipulación, repudio, divulgación de información, denegación de servicio y elevación de privilegios— para que una revisión sea sistemática.
ubicación de controles	Elegir el límite en el que se puede hacer cumplir una regla. Una regla que solo existe en la interfaz es una sugerencia; la misma regla dentro de un servicio es un control.

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 1. Puedes abrir tú mismo [module-01.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 1 – «El modelo de seguridad de Wisej.NET» – 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-01.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: Crea la solución AccessOps y redacta su modelo de amenazas: dibuja los límites de confianza, enumera los activos (fichas de empleados, documentos, aprobaciones), identifica los puntos de entrada (inicio de sesión, subidas, descargas, endpoints personalizados, el asistente de IA), valora cada amenaza y asigna cada mitigación al módulo de este curso que la implementa. Construye el esqueleto de la aplicación con una página de inicio de sesión provisional.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Solución AccessOps con una página de inicio de sesión provisional
 - Diagrama de límites de confianza
 - Inventario de activos y puntos de entrada
 - Tabla de amenazas con probabilidad, impacto y mitigación
 - Mapa de ubicación de los controles de seguridad
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 cuadrícula de empleados está enlazada a las fichas completas y la columna del salario está oculta en el diseñador. Describe lo que tiene el navegador y nombra el cambio que de verdad saca el salario de la red.
- Dos personas tienen la sesión iniciada en AccessOps en el mismo servidor. Explica qué partes del estado comparten por defecto y qué tendría que ocurrir para que una de ellas viera el departamento actual de la otra.
- Elige la amenaza de tu tabla con mayor impacto y defiende la postura contraria: ¿qué tendría que ser cierto en AccessOps para que esa mitigación fuera una pérdida de tiempo para el equipo?

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 cuadrícula de empleados está enlazada a las fichas completas y la columna del salario está oculta en el diseñador. Describe lo que tiene el navegador y nombra el cambio que de verdad saca el salario de la red.
- ✓ Dos personas tienen la sesión iniciada en AccessOps en el mismo servidor. Explica qué partes del estado comparten por defecto y qué tendría que ocurrir para que una de ellas viera el departamento actual de la otra.
- ✓ Elige la amenaza de tu tabla con mayor impacto y defiende la postura contraria: ¿qué tendría que ser cierto en AccessOps para que esa mitigación fuera una pérdida de tiempo para el equipo?

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-01.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."*
- ✓ **Revisalo como un pull request.** *"Explicame 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.