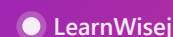


Ejercicio de programación con IA

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.



El asistente pone las manos rápidas. El arquitecto sigues siendo tú.

Un asistente puede generar el ejemplo de este módulo en minutos. Lo que no puede hacer es decidir si el resultado es el diseño adecuado para tu aplicación — esa decisión sigue siendo tuya. Tú pones el criterio: dale unas buenas instrucciones y luego contrasta lo que te devuelva con lo que te acaba de enseñar el Módulo 7.

Qué estás pidiendo

Completa el proyecto final de AccessOps: impón HTTPS con HSTS y configura las cabeceras de seguridad, marca las cookies como Secure y HttpOnly con una política SameSite estricta, fija el tiempo de expiración de la sesión y los límites de tamaño de las solicitudes, añade limitación de frecuencia al inicio de sesión y a los enlaces compartidos, ejecuta una auditoría de dependencias y registra los resultados, y después recorre la lista de comprobación completa de este curso contra la aplicación y corrige cada hallazgo.

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

- ✓ Configuración de HTTPS, HSTS y cabeceras de seguridad
- ✓ Ajustes de bastionado de cookies y sesión
- ✓ Limitación de frecuencia en el inicio de sesión y los enlaces compartidos
- ✓ Informe de auditoría de dependencias
- ✓ Lista de comprobación de la revisión de seguridad completada con las correcciones

Por qué tiene esta forma

AccessOps ya verifica las contraseñas contra un hash con sal, resuelve los permisos en un servicio, transmite los documentos solo a través de DocumentService y audita cada aprobación. Nada de eso sirve si la primera solicitud viaja en texto claro, si un script puede leer la cookie de sesión o si el build incluye un paquete con un aviso de seguridad publicado. El último tramo corresponde al host: el certificado, las cabeceras de respuesta, los atributos de las cookies, los límites de las solicitudes y la versión de todo lo que hay en el grafo de dependencias.

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
HSTS	Una cabecera Strict-Transport-Security que indica al navegador que rechace HTTP sin cifrar para este host durante un tiempo determinado, para que una visita posterior nunca pueda interceptarse en la redirección.
Content-Security-Policy	Una cabecera que enumera los orígenes desde los que una página puede cargar scripts, estilos y marcos; convierte un script inyectado en una solicitud bloqueada en lugar de código ejecutándose en una sesión activa.
frame-ancestors	La directiva CSP (X-Frame-Options es el equivalente antiguo) que impide que otro sitio incruste AccessOps en un marco y engañe a un responsable para que pulse Approve.
atributos de la cookie	Secure mantiene la cookie de sesión fuera de HTTP sin cifrar, HttpOnly la mantiene lejos de JavaScript y SameSite impide que viaje en una solicitud iniciada por otro sitio.
límites de las solicitudes	Topes a nivel de host en el tamaño del cuerpo y de las cabeceras que rechazan una subida sobredimensionada antes de que se ejecute una sola línea de código de AccessOps.
token bucket	Un contador que concede un intento solo mientras contiene un token y se rellena a un ritmo fijo, de modo que el tráfico puede tener un pequeño pico y después se ralentiza al mínimo.
auditoría de dependencias	Una comprobación repetible de cada paquete directo y transitivo, y del runtime, contra los avisos de seguridad publicados, registrada con una fecha y una decisión.

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 7. Puedes abrir tú mismo [module-07.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 7 – «Lista de comprobación de bastionado y proyecto final» – 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-07.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: Completa el proyecto final de AccessOps: impón HTTPS con HSTS y configura las cabeceras de seguridad, marca las cookies como Secure y HttpOnly con una política SameSite estricta, fija el tiempo de expiración de la sesión y los límites de tamaño de las solicitudes, añade limitación de frecuencia al inicio de sesión y a los enlaces compartidos, ejecuta una auditoría de dependencias y registra los resultados, y después recorre la lista de comprobación completa de este curso contra la aplicación y corrige cada hallazgo.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Configuración de HTTPS, HSTS y cabeceras de seguridad
 - Ajustes de bastionado de cookies y sesión
 - Limitación de frecuencia en el inicio de sesión y los enlaces compartidos
 - Informe de auditoría de dependencias
 - Lista de comprobación de la revisión de seguridad completada con las correcciones
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 middleware puede limitar la frecuencia del handler de enlaces compartidos, pero no la del botón Sign in. ¿Qué diferencia hay entre ambas solicitudes en una aplicación Wisej.NET, y dónde tiene que vivir entonces el límite del inicio de sesión?
- Tu Content-Security-Policy está en modo obligatorio y la cuadrícula se muestra sin estilos. ¿Qué directiva es la causa probable, y cómo te lo habría advertido el modo report-only antes de que lo hiciera el despliegue?
- La auditoría de dependencias muestra un paquete vulnerable al que nadie en AccessOps hace referencia directa. ¿Cómo llegó al build, y cuáles son las dos formas honestas de cerrar el hallazgo?

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 middleware puede limitar la frecuencia del handler de enlaces compartidos, pero no la del botón Sign in. ¿Qué diferencia hay entre ambas solicitudes en una aplicación Wisej.NET, y dónde tiene que vivir entonces el límite del inicio de sesión?
- ✓ Tu Content-Security-Policy está en modo obligatorio y la cuadrícula se muestra sin estilos. ¿Qué directiva es la causa probable, y cómo te lo habría advertido el modo report-only antes de que lo hiciera el despliegue?
- ✓ La auditoría de dependencias muestra un paquete vulnerable al que nadie en AccessOps hace referencia directa. ¿Cómo llegó al build, y cuáles son las dos formas honestas de cerrar el hallazgo?

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.

- ✓ **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."

- ✓ **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."*



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.