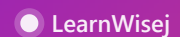


Ejercicio de programación con IA

Construye el ejemplo EnterpriseOps 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 — incluso comparándolo con nuestra propia solución de referencia en GitHub.



Escribir el código ya es la mitad fácil.

Conseguir que un asistente moderno genere código que funcione para «JavaScript Object Model, Browser APIs, and Secure Interop Contracts» 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 una paleta de comandos de teclado avanzada y un panel de capacidades del navegador. Usa JavaScript para recopilar las funciones del navegador y lanzar comandos en el servidor, y después aplica los permisos en el servidor antes de ejecutar el comando.

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

- ✓ Documento del contrato de interoperabilidad
- ✓ Script de la paleta de comandos
- ✓ Métodos de callback del servidor
- ✓ Panel de capacidades del navegador
- ✓ Notas de la revisión de seguridad

Por qué tiene esta forma

Wisej.NET permite a los desarrolladores añadir JavaScript allí donde aporta valor, pero la interoperabilidad avanzada puede volverse frágil a menos que cada límite se trate como una pequeña API.

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 objetos App	La representación del lado del cliente de los componentes de Wisej.NET a la que puede acceder JavaScript. Úsala de forma deliberada, a través de puntos de entrada documentados, sin hurgar en los detalles internos.
evento del cliente	Un evento del lado del navegador que se reenvía al servidor. Solo los eventos significativos deberían viajar por la red, con cargas útiles compactas.
método remoto	Un método de servidor al que el navegador puede llamar directamente, como un WebMethod, que devuelve un resultado a JavaScript. Necesita la misma validación que cualquier endpoint.
detección de funciones del navegador	Comprobar qué admite el navegador antes de usar una API. Úsala solo para aportar valor en el navegador, mientras las reglas de negocio permanecen en el servidor.
contrato de interoperabilidad	Una descripción documentada y con nombre de lo que JavaScript envía al servidor y de lo que recibe de vuelta, para que ambos lados se puedan probar y versionar.
momento del ciclo de vida	Saber cuándo se crea, se renderiza, se actualiza y se destruye un widget de cliente, para que los scripts se ejecuten en el momento adecuado y limpien lo que dejan.
validación de entradas	Tratar cada valor que llega desde el navegador como no confiable y validarlo en el servidor, porque cualquiera puede modificar el código del cliente.

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 9. Puedes abrir tú mismo [module-09.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 EnterpriseOps 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 9 – «JavaScript Object Model, Browser APIs, and Secure Interop Contracts» – del curso de LearnWisej «Enterprise Wisej.NET: Architecture to Cloud», 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/azure-docker/specs/module-09.md>

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

1. Proyecto «EnterpriseOps», 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 una paleta de comandos de teclado avanzada y un panel de capacidades del navegador. Usa JavaScript para recopilar las funciones del navegador y lanzar comandos en el servidor, y después aplica los permisos en el servidor antes de ejecutar el comando.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Documento del contrato de interoperabilidad
 - Script de la paleta de comandos
 - Métodos de callback del servidor
 - Panel de capacidades del navegador
 - Notas de la revisión 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:

- ¿Qué ocurre antes de que exista el widget de destino?
- ¿En qué valores procedentes de JavaScript se confía?
- ¿Qué reglas de negocio permanecen en el servidor?

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:

- ✓ ¿Qué ocurre antes de que exista el widget de destino?
- ✓ ¿En qué valores procedentes de JavaScript se confía?
- ✓ ¿Qué reglas de negocio permanecen en el servidor?

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.



Este ya lo construimos — compara tu versión con la nuestra

La implementación de referencia del Módulo 9 está en el repositorio público de ejemplos. Construye primero la tuya y después abre la nuestra: lo interesante son las diferencias.



[Abrir el ejemplo del Módulo 9 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Enterprise Wisej.NET Course/Module 9`.



Nuestro ejemplo es un punto de partida, no un producto terminado

Se construyó igual que estás construyendo el tuyo — con un prompt y luego corregido — y se detiene a propósito donde se detiene el módulo. Espera soluciones provisionales donde iría código de producción, un manejo de errores mínimo y ninguna prueba. Tómallo como una invitación: usa lo que aprendiste en este módulo, junto con el asistente, para llevarlo más lejos. Mejorar código que no escribiste tú es precisamente la habilidad que enseña este curso.

Tres prompts para ir más allá

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

- ✓ **Añade el segundo caso.** *"Amplía este proyecto EnterpriseOps 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/azure-docker/specs/module-09.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."*
- ✓ **Revísalo como un pull request.** *"Explícame este código de EnterpriseOps 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."*



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.