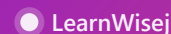


Ejercicio de programación con IA

Construye el ejemplo AdaptiveOps 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.



Instrúyelo, léelo y luego rómpelo.

Este ejercicio tiene tres partes, y solo la primera trata del prompt. Le das al asistente la especificación del módulo y recibes una solución AdaptiveOps; lees lo que escribió y decides si de verdad hace lo que pedía el módulo; y después lo llevas más allá de lo que pedía el módulo. En la segunda y la tercera parte es donde se aprende.

Qué estás pidiendo

Añade a Adaptive Operations Console una hoja de estilos AdaptiveOps.css con ámbito, con clases para tarjetas de métricas y badges compactos, aplícalas mediante CssClass y usa CssStyle solo para un valor calculado, como el ancho de una barra de progreso. Asigna la AppearanceKey action-button al comando principal, añade un estado personalizado stale a las filas de la cuadrícula de tickets o a un panel de estado, e implementa en la barra de herramientas un cambio de tema claro/oscuro que modifique el tema solo para la sesión actual, sin alterar el tema compartido.

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

- ✓ AdaptiveOps.css con ámbito y las clases metric-card y compact-badge aplicadas mediante CssClass
- ✓ CssStyle usado exactamente para un valor calculado puntual, con el motivo documentado
- ✓ Comando principal con la AppearanceKey action-button
- ✓ Estado personalizado stale añadido mediante States y con estilo definido en el tema
- ✓ Cambio de tema claro/oscuro solo para la sesión que deja intacto el tema compartido

Por qué tiene esta forma

El tema AdaptiveOps del Módulo 2 ya es responsable de la identidad de todos los controles estándar de Adaptive Operations Console, y las peticiones que llegan a continuación son todas pequeñas: una tarjeta de métricas necesita una superficie redondeada y elevada, un badge de SLA debe ser compacto, una barra de progreso necesita un ancho calculado a partir de datos en vivo, el comando Save debe parecer la acción principal, y el turno de noche quiere una variante oscura. Cada petición tiene un atajo tentador — aquí una cadena CssStyle, allí un BackColor, el tema compartido editado directamente— y cada atajo traslada en silencio una decisión visual a una capa que no es su responsable. Seis meses después nadie

sabe decir qué renderiza el tema, qué sobrescribe el CSS y qué parchea el código. Este módulo coloca cada una de esas cinco peticiones en el mecanismo responsable.

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
CssClass	Uno o varios nombres de clase añadidos al elemento principal del widget en el navegador: el punto de enganche reutilizable al que puede apuntar una hoja de estilos de la aplicación con ámbito sin tocar el tema.
CssStyle	Una cadena CSS que se escribe directamente como atributo style del elemento principal del widget; adecuada para un único valor calculado, inadecuada como herramienta de estilo general porque los estilos en línea son difíciles de inspeccionar, de tematizar y de entregar a un diseñador.
hoja de estilos de la aplicación con ámbito	Un archivo CSS cargado junto al tema cuyos selectores parten de nombres de clase propios, como.metric-card, y nunca entran en el DOM interno de un widget estándar de Wisej.
extender heredado StyleSheet	Un componente antiguo que añade clases CSS a los controles compatibles y aloja reglas en su propiedad StyleSheetSource; una ayuda para la migración, innecesaria en código nuevo porque todos los controles tienen ya CssClass y CssStyle.
AppearanceKey	El nombre de una appearance del tema activo con la que se renderiza un control, de modo que el C# expresa la intención (action-button) y el tema es responsable de los colores, fuentes y métricas de cada estado.
estados personalizados	Nombres añadidos a la colección States de un control a los que el tema puede dar estilo como a hovered o focused, para condiciones transitorias de la interfaz como stale, warning o dirty que un diseñador debe poder ver y rediseñar.
tema de arranque frente a Application.Theme	El tema de toda la aplicación se elige antes de que se ejecute en Web.config o Default.json; Application.Theme permite que una sesión trabaje con un objeto de tema en tiempo de ejecución sin cambiar lo que ven las demás sesiones.

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 3. Puedes abrir tú mismo [module-03.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



Usar Claude

Abre [claude.ai](#) con el mismo prompt ya cargado. Si usas Claude Code, puedes

Codex; cualquier plan que pueda navegar por la web descargará la especificación.

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 AdaptiveOps 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 3 – «CSS, CssClass, CssStyle, estados y cambios de tema en tiempo de ejecución» – del curso de LearnWisej «Temas y diseños adaptables», 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/theming-and-responsive-layouts/specs/module-03.md>

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

1. Proyecto «AdaptiveOps», 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 a Adaptive Operations Console una hoja de estilos AdaptiveOps.css con ámbito, con clases para tarjetas de métricas y badges compactos, aplícalas mediante `CssClass` y usa `CssStyle` solo para un valor calculado, como el ancho de una barra de progreso. Asigna la `AppearanceKey` `action-button` al comando principal, añade un estado personalizado `stale` a las filas de la cuadrícula de tickets o a un panel de estado, e implementa en la barra de herramientas un cambio de tema claro/oscuro que modifique el tema solo para la sesión actual, sin alterar el tema compartido.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - AdaptiveOps.css con ámbito y las clases `metric-card` y `compact-badge` aplicadas mediante `CssClass`
 - `CssStyle` usado exactamente para un valor calculado puntual, con el motivo documentado
 - Comando principal con la `AppearanceKey` `action-button`
 - Estado personalizado `stale` añadido mediante `States` y con estilo definido en el tema
 - Cambio de tema claro/oscuro solo para la sesión que deja intacto el tema compartido
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 diseñador pide un radio de esquina más pequeño en todas las tarjetas de métricas. ¿Qué archivo cambia, y cuántos lugares habrías tenido que editar si el radio se hubiera asignado con `CssStyle`?
- ¿Por qué el botón Assign no necesita código de hover ni de pressed, y qué perderías exactamente al sustituir su `AppearanceKey` por valores de `BackColor` y `ForeColor` que coincidan con la paleta?
- Hay dos personas conectadas y una de ellas cambia al modo oscuro. Describe lo que ve cada una y señala la línea de tu código que lo garantiza.

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 diseñador pide un radio de esquina más pequeño en todas las tarjetas de métricas. ¿Qué archivo cambia, y cuántos lugares habrías tenido que editar si el radio se hubiera asignado con `CssStyle`?
- ✓ ¿Por qué el botón Assign no necesita código de hover ni de pressed, y qué perderías exactamente al sustituir su `AppearanceKey` por valores de `BackColor` y `ForeColor` que coincidan con la paleta?
- ✓ Hay dos personas conectadas y una de ellas cambia al modo oscuro. Describe lo que ve cada una y señala la línea de tu código que lo garantiza.

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 3 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 3 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Theming and Responsive Layouts Course/Module 3`.



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.

- ✓ **Revísalo como un pull request.** *"Explicame este código de AdaptiveOps 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 AdaptiveOps 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 AdaptiveOps 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.