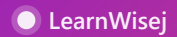


Ejercicio de programación con IA

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.



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 2.

Qué estás pidiendo

Crea el tema AdaptiveOps a partir de un tema base integrado en el Theme Builder. Define una paleta de colores con nombre con brandPrimary, brandAccent, surface, surfaceAlt, textMain, textMuted, danger, warning, success y focusFrame, además de las fuentes default, heading y mono, y luego rediseña las appearances de botón, panel, pestaña, encabezado de cuadrícula, editor no válido y tooltip usando esos tokens. Añade una appearance action-button que herede de button y sobrescriba solo sus estados default, hovered y pressed, verifica cada cambio en la vista previa del Theme Builder y carga el tema en Adaptive Operations Console.

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

- ✓ AdaptiveOps.theme creado a partir de un tema base y cargado por la aplicación
- ✓ Tokens de color con nombre y tokens de fuente default, heading y mono definidos una vez y reutilizados
- ✓ Appearances de botón, panel, pestaña, encabezado de cuadrícula, editor no válido y tooltip rediseñadas a partir de los tokens
- ✓ Appearance action-button que hereda de button con los estados default, hovered y pressed sobrescritos
- ✓ Capturas de la vista previa del Theme Builder y una nota sobre las decisiones de orden de estados y herencia

Por qué tiene esta forma

El Módulo 1 dio a Adaptive Operations Console un shell acoplado, pero el tema base sigue siendo responsable de todos los colores y fuentes. El atajo es aplicar la marca control por control con BackColor, Font, cadenas CssStyle o CSS contra el DOM interno de los widgets. Cada uno de ellos dispersa un mismo valor hexadecimal por decenas de sitios, lucha contra el motor de temas que ya renderiza todos los

estados y se rompe en la siguiente actualización. Un tema es un único archivo JSON responsable de la identidad de los controles estándar; el Theme Builder es la herramienta para producirlo y mantenerlo, y este módulo construye con él el tema AdaptiveOps.

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
tema base	El tema integrado que el Theme Builder copia para crear el tuyo, de modo que cientos de appearances, componentes, estados, métricas, imágenes y fuentes existen antes de que cambies un solo valor.
tokens con nombre de color y fuente	Entradas de las secciones colors y fonts (brandPrimary, danger, heading) que las appearances referencian por nombre, de modo que un cambio de marca es una sola edición.
appearance y AppearanceKey	Una appearance es el bloque del tema para un tipo de widget; la AppearanceKey de un control selecciona qué bloque se aplica, y así existe una variante semántica como un botón de acción.
herencia de appearances	Una entrada inherit permite que una appearance especializada reutilice otra base y sobrescriba solo unos pocos estados, de modo que action-button sigue siendo un botón real en todo lo demás.
orden de los estados	Estados como default, hovered, pressed, focused, disabled e invalid se aplican en el orden de la lista; mover uno cambia qué estilos ganan, así que el orden es comportamiento.
componente hijo	Una parte interna de un widget, como la barra de título de un panel, el encabezado de columna de una cuadrícula o el botón de un editor, con su propia ruta de appearance y sus propios estados.
estilos frente a propiedades	Los estilos son valores de tipo CSS que el tema convierte en clases CSS dinámicas y de los que es responsable; las propiedades son propiedades del widget que pueden convertirse en estilo en línea o en comportamiento y que la aplicación puede sobrescribir.

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 2. Puedes abrir tú mismo [module-02.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 2 – «Theme Builder y el funcionamiento interno del JSON de temas» – 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-02.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: Crea el tema AdaptiveOps a partir de un tema base integrado en el Theme Builder. Define una paleta de colores con nombre con `brandPrimary`, `brandAccent`, `surface`, `surfaceAlt`, `textMain`, `textMuted`, `danger`, `warning`, `success` y `focusFrame`, además de las fuentes `default`, `heading` y `mono`, y luego rediseña las appearances de botón, panel, pestaña, encabezado de cuadrícula, editor no válido y tooltip usando esos tokens. Añade una appearance action-button que herede de `button` y sobrescriba solo sus estados `default`, `hovered` y `pressed`, verifica cada cambio en la vista previa del Theme Builder y carga el tema en Adaptive Operations Console.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - AdaptiveOps.theme creado a partir de un tema base y cargado por la aplicación
 - Tokens de color con nombre y tokens de fuente `default`, `heading` y `mono` definidos una vez y reutilizados
 - Appearances de botón, panel, pestaña, encabezado de cuadrícula, editor no válido y tooltip rediseñadas a partir de los tokens
 - Appearance action-button que hereda de `button` con los estados `default`, `hovered` y `pressed` sobrescritos
 - Capturas de la vista previa del Theme Builder y una nota sobre las decisiones de orden de estados y herencia
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:

- Mañana marketing cambia `brandPrimary` por un nuevo valor hexadecimal. ¿Qué archivos y qué líneas cambian en tu solución, y cuál habría sido la respuesta si hubieras asignado `BackColor` a cada botón?
- Has movido el estado `pressed` por encima de `hovered` en la appearance action-button. ¿Qué ve el usuario mientras mantiene pulsado el botón del ratón, y por qué lo decide el orden y no

el color?

– Un TextBox del editor de detalles se queda gris cuando no es válido aunque diste estilo al estado invalid. ¿Qué tres cosas compruebas, y en qué orden, antes de volver a editar el tema?

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:

- ✓ Mañana marketing cambia brandPrimary por un nuevo valor hexadecimal. ¿Qué archivos y qué líneas cambian en tu solución, y cuál habría sido la respuesta si hubieras asignado BackColor a cada botón?
- ✓ Has movido el estado pressed por encima de hovered en la appearance action-button. ¿Qué ve el usuario mientras mantiene pulsado el botón del ratón, y por qué lo decide el orden y no el color?
- ✓ Un TextBox del editor de detalles se queda gris cuando no es válido aunque diste estilo al estado invalid. ¿Qué tres cosas compruebas, y en qué orden, antes de volver a editar el tema?

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 2 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 2 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Theming and Responsive Layouts Course/Module 2`.



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.

- ✓ **Demuestra que de verdad funciona.** *"Recorre este proyecto entregable por entregable frente a la especificación del módulo en <https://learnwisej.com/assets/courses/theming-and-responsive-layouts/specs/module-02.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.** *"Explícame 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."*



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.