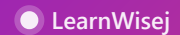


Ejercicio de programación con IA

Construye el ejemplo VisualOperationsStudio 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

Añade a VisualOperationsStudio un TopologyImageRenderer que convierta el modelo de topología en un PNG sin tocar ningún control de Wisej. Dale un método Render que reciba la escena y un tamaño de salida, cree el Bitmap, obtenga su Graphics con Graphics.FromImage, dibuje nodos, aristas, etiquetas y una leyenda con pens, brushes y un GraphicsPath, y libere todo lo que le pertenece sin tocar los objetos de quien lo llama. Resuelve su Font mediante un método ResolveFont que pruebe una familia proporcionada por la aplicación y recurra a otra que sepas presente, en lugar de dar por hecho que existe una fuente del sistema; guarda el resultado como PNG en un MemoryStream y entrega los bytes a la página para su descarga. Después revisa las referencias del proyecto para confirmar que no estás añadiendo System.Drawing.Common junto a la implementación gestionada de Wisej, ejecuta o prepara el renderizador en Windows y en un contenedor Linux, y documenta cómo se comporta el mismo código en destinos iOS, Android y macOS Hybrid.

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

- ✓ TopologyImageRenderer sin dependencia de ningún control de Wisej, que dibuja con Graphics.FromImage sobre un Bitmap
- ✓ Método ResolveFont con una cadena de alternativas explícita en lugar de una fuente del sistema dada por supuesta
- ✓ PNG escrito en un MemoryStream y descargado desde la página, con todos los objetos gráficos propios liberados
- ✓ Comprobación de referencias que confirme que se usa Managed.System.Drawing y que System.Drawing.Common no se añade junto a él
- ✓ CrossPlatform.md con evidencias en Windows y en un contenedor Linux, y las implicaciones para iOS, Android y macOS

Por qué tiene esta forma

Todo lo que VisualOperationsStudio ha dibujado hasta ahora pertenece a una sesión viva: el indicador se redibuja porque un control lo ha pedido, y la topología en Canvas del Módulo 5 existe solo como comandos que el navegador ya ha ejecutado. Operaciones quiere ahora esa misma topología en un correo nocturno de incidencias y adjunta a un ticket, lo que significa una imagen sin ningún control detrás. Al mismo tiempo, la aplicación se está moviendo a un contenedor Linux, así que el código de dibujo tiene que seguir funcionando en un host sin GDI+ y sin libgdiplus.

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
System.Drawing.Common	El envoltorio de GDI+ de Microsoft, específico de Windows desde .NET 6; el conmutador para Unix desapareció en .NET 7, así que no es una opción para Linux, macOS, iOS ni Android.
Managed.System.Drawing	El identificador del paquete NuGet, invertido porque la raíz de paquetes System.* está reservada; Wisej.NET 4 ya lo referencia por ti.
System.Drawing.Managed.dll	El ensamblado que entrega ese paquete: la propia implementación gestionada, que se ve en un registro de compilación y no en tu código fuente.
El espacio de nombres System.Drawing	Sin cambios: Bitmap, Graphics, Pen y Font conservan sus nombres, así que el código de dibujo existente compila sin reescribirlo.
Motor de gráficos gestionado	Una implementación totalmente gestionada sobre un motor ImageSharp personalizado e integrado; en el .NET moderno evita GDI+, libgdiplus y las llamadas nativas.
Graphics.FromImage	El punto de entrada fuera de pantalla: una superficie de dibujo sobre un Bitmap que te pertenece, sin control, sin evento Paint y sin sesión.
PrivateFontCollection	Carga los archivos de fuente que distribuye la aplicación, de modo que la salida depende de lo que hayas desplegado y no de lo que resulte estar instalado en el host.

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 6. Puedes abrir tú mismo [module-06.md](#) para ver exactamente lo que leerá.



Usar ChatGPT



Usar Claude

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.

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 VisualOperationsStudio 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 6 – «System.Drawing.Managed en Windows, Linux, macOS, iOS y Android» – del curso de LearnWisej «Dibujo y pintura en Wisej.NET», 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/drawing-and-painting/specs/module-06.md>

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

1. Proyecto «VisualOperationsStudio», 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 `VisualOperationsStudio` un `TopologyImageRenderer` que convierta el modelo de topología en un PNG sin tocar ningún control de Wisej. Dale un método `Render` que reciba la escena y un tamaño de salida, cree el `Bitmap`, obtenga su `Graphics` con `Graphics.FromImage`, dibuje nodos, aristas, etiquetas y una leyenda con pens, brushes y un `GraphicsPath`, y libere todo lo que le pertenece sin tocar los objetos de quien lo llama. Resuelve su `Font` mediante un método `ResolveFont` que pruebe una familia proporcionada por la aplicación y recurra a otra que sepas presente, en lugar de dar por hecho que existe una fuente del sistema; guarda el resultado como PNG en un `MemoryStream` y entrega los bytes a la página para su descarga. Después revisa las referencias del proyecto para confirmar que no estás añadiendo `System.Drawing.Common` junto a la implementación gestionada de Wisej, ejecuta o prepara el renderizador en Windows y en un contenedor Linux, y documenta cómo se comporta el mismo código en destinos iOS, Android y macOS Hybrid.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `TopologyImageRenderer` sin dependencia de ningún control de Wisej, que dibuja con `Graphics.FromImage` sobre un `Bitmap`
 - Método `ResolveFont` con una cadena de alternativas explícita en lugar de una fuente del sistema dada por supuesta
 - PNG escrito en un `MemoryStream` y descargado desde la página, con todos los objetos gráficos propios liberados
 - Comprobación de referencias que confirme que se usa `Managed.System.Drawing` y que `System.Drawing.Common` no se añade junto a él
 - `CrossPlatform.md` con evidencias en Windows y en un contenedor Linux, y las implicaciones para iOS, Android y macOS
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 compañero dice que el renderizador demuestra que la aplicación es multiplataforma porque el código compila en Linux. ¿Qué tendrías que observar realmente en el contenedor antes de aceptar esa afirmación, y qué parte de la salida es más probable que difiera?
- Tu renderizador libera el Bitmap que ha creado. ¿Por qué liberar el Graphics dentro del controlador de Paint de un control sería un error, y qué regla cubre ambos casos en una sola frase?
- Alguien propone sustituir este renderizador por SkiaSharp porque en las pruebas de rendimiento es más rápido. ¿Qué necesitarías saber sobre la carga de trabajo antes de aceptarlo, y qué pierde el equipo al abandonar la API de System.Drawing?

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 compañero dice que el renderizador demuestra que la aplicación es multiplataforma porque el código compila en Linux. ¿Qué tendrías que observar realmente en el contenedor antes de aceptar esa afirmación, y qué parte de la salida es más probable que difiera?
- ✓ Tu renderizador libera el Bitmap que ha creado. ¿Por qué liberar el Graphics dentro del controlador de Paint de un control sería un error, y qué regla cubre ambos casos en una sola frase?
- ✓ Alguien propone sustituir este renderizador por SkiaSharp porque en las pruebas de rendimiento es más rápido. ¿Qué necesitarías saber sobre la carga de trabajo antes de aceptarlo, y qué pierde el equipo al abandonar la API de System.Drawing?

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.

- ✓ **Añade el segundo caso.** *"Amplía este proyecto VisualOperationsStudio 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/drawing-and-painting/specs/module-06.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 VisualOperationsStudio 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.