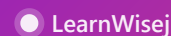


Ejercicio de programación con IA

Construye el ejemplo GlobalDesk 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

Termina el proyecto final de GlobalDesk. Envuelve toda búsqueda de recursos en un `LocalizationService` con métodos `Text`, `Date`, `Number` y `Currency`, dale una política de claves ausentes que muestre la clave entre corchetes en desarrollo y la registre en producción, y haz que ese servicio sea el único sitio donde la interfaz toca `ResourceManager`. Mantén el dominio al margen: el modelo del ticket expone un valor `TicketStatus` y la interfaz lo asigna a una etiqueta localizada. Después ejecuta la matriz de pruebas de localización sobre la aplicación terminada, cubriendo la cultura neutra, el alemán, una cultura de idioma y región que se repliega, una clave ausente a propósito, fechas e importes formateados, un cambio en tiempo de ejecución, una pasada con cadenas largas y la cultura RTL, usando un conjunto de recursosseudolocalizado para sacar a la luz todo lo que siga escrito en el código. Confirma que las carpetas de cultura sobreviven a una publicación y entrega la lista de comprobación de QA como entregable del proyecto final.

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

- ✓ `LocalizationService` con métodos `Text`, `Date`, `Number` y `Currency` y una política documentada para las claves ausentes
- ✓ Código de dominio que devuelve un valor `TicketStatus` que la interfaz asigna a una etiqueta localizada
- ✓ Conjunto de recursosseudolocalizado usado para encontrar todas las cadenas que sigan escritas en el código
- ✓ Matriz de pruebas de localización completada, que cubre el repliegue, el formato, el cambio en tiempo de ejecución y RTL
- ✓ Salida publicada comprobada en busca de sus carpetas de cultura, más la lista de comprobación de QA de localización de GlobalDesk

Por qué tiene esta forma

Después de seis módulos GlobalDesk habla dos idiomas, pero la maquinaria está dispersa: recursos de diseñador en el editor de clientes, un helper sobre los archivos.resx compartidos, un selector de idioma, un manejador de CultureChanged y una demostración de RTL. Todavía nada declara qué ocurre cuando una clave no tiene traducción, y nadie ha demostrado que la compilación en alemán siga cabiendo en una ventana estrecha. Un sistema de localización de producción es predecible: sabe de dónde viene cada cadena, qué cultura está activa en esta sesión, qué aspecto tiene una traducción ausente y qué le hace un cambio de idioma a las pantallas abiertas. Este módulo convierte esas cuatro respuestas en un pequeño servicio, una matriz de pruebas escrita y el proyecto final terminado.

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
LocalizationService	Una clase pequeña que centraliza las decisiones de tu aplicación alrededor de la localización de.NET en lugar de sustituirla: un nombre base de recursos, una política de claves ausentes, un conjunto de métodos de formato.
Política de claves ausentes	El comportamiento elegido y probado cuando GetString no devuelve nada: un marcador [Key] visible mientras desarrollas, y la clave registrada más el texto neutro de repliegue en producción.
Texto de producto, de sistema y de diagnóstico	Tres familias con tres propietarios: tus propias cadenas.resx, las etiquetas propias de Wisej.NET a través de Resources-[LANG].resx, y el texto de registros o excepciones que se queda en un único idioma para desarrolladores.
Dominio sin cultura	El modelo expone un valor como TicketStatus.Closed y la interfaz lo asigna a una etiqueta localizada, de modo que la lógica de negocio y sus pruebas nunca dependen de un idioma.
Matriz de pruebas de localización	Una lista escrita de ejecuciones (neutra, traducida, con repliegue de región, clave ausente, valores formateados, cambio en tiempo de ejecución, cadenas largas, RTL, etiquetas de sistema) para que la cobertura sea una lista de comprobación y no un recuerdo.
Seudolocalización	Un conjunto de recursos de prueba cuyos valores están visiblemente alterados y alargados, como [!!! Save customer record!!!], creado para sacar a la luz los recortes y las cadenas escritas en el código, no para traducir nada.
Recursos satélite	.NET compila los recursos específicos de cultura en ensamblados satélite, así que las carpetas de cultura que produce la compilación tienen que viajar con la aplicación cuando la publicas o la empaquetas.

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 GlobalDesk 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 – «Arquitectura de localización en producción, pruebas y proyecto final» – del curso de LearnWisej «Localización 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/localization-in-wisej-net/specs/module-07.md>

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

1. Proyecto «GlobalDesk», 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: Termina el proyecto final de GlobalDesk. Envuelve toda búsqueda de recursos en un `LocalizationService` con métodos `Text`, `Date`, `Number` y `Currency`, dale una política de claves ausentes que muestre la clave entre corchetes en desarrollo y la registre en producción, y haz que ese servicio sea el único sitio donde la interfaz toca `ResourceManager`. Mantén el dominio al margen: el modelo del ticket expone un valor `TicketStatus` y la interfaz lo asigna a una etiqueta localizada. Después ejecuta la matriz de pruebas de localización sobre la aplicación terminada, cubriendo la cultura neutra, el alemán, una cultura de idioma y región que se repliega, una clave ausente a propósito, fechas e importes formateados, un cambio en tiempo de ejecución, una pasada con cadenas largas y la cultura RTL, usando un conjunto de recursos seudolocalizado para sacar a la luz todo lo que siga escrito en el código. Confirma que las carpetas de cultura sobreviven a una publicación y entrega la lista de comprobación de QA como entregable del proyecto final.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `LocalizationService` con métodos `Text`, `Date`, `Number` y `Currency` y una política documentada para las claves ausentes
 - Código de dominio que devuelve un valor `TicketStatus` que la interfaz asigna a una etiqueta localizada
 - Conjunto de recursos seudolocalizado usado para encontrar todas las cadenas que sigan escritas en el código
 - Matriz de pruebas de localización completada, que cubre el repliegue, el formato, el cambio en tiempo de ejecución y RTL
 - Salida publicada comprobada en busca de sus carpetas de cultura, más la lista de comprobación de QA de localización de GlobalDesk
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 revisor abre GlobalDesk en francés, una cultura que nunca tradujiste, y ve etiquetas en inglés con fechas francesas correctamente formateadas. Explica qué mecanismo produjo cada mitad de esa pantalla y di si hay algo roto.
- Tu ejecución seudolocalizada muestra una etiqueta que sigue poniendo "Save" en inglés a secas. Enumera los tres sitios de los que podría venir ese texto y describe cómo los distinguirías sin adivinar.
- Un compañero propone guardar en caché el CultureInfo resuelto en un campo estático de LocalizationService "porque buscarlo en cada llamada es un derroche". Describe qué experimentaría un segundo usuario con la sesión iniciada y qué propondrías tú en su lugar.

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 revisor abre GlobalDesk en francés, una cultura que nunca tradujiste, y ve etiquetas en inglés con fechas francesas correctamente formateadas. Explica qué mecanismo produjo cada mitad de esa pantalla y di si hay algo roto.
- ✓ Tu ejecución seudolocalizada muestra una etiqueta que sigue poniendo "Save" en inglés a secas. Enumera los tres sitios de los que podría venir ese texto y describe cómo los distinguirías sin adivinar.
- ✓ Un compañero propone guardar en caché el CultureInfo resuelto en un campo estático de LocalizationService "porque buscarlo en cada llamada es un derroche". Describe qué experimentaría un segundo usuario con la sesión iniciada y qué propondrías tú en su lugar.

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.

- ✓ **Demuestra que de verdad funciona.** "Recorre este proyecto entregable por entregable frente a la especificación del módulo en <https://learnwisej.com/assets/courses/localization-in-wisej-net/specs/module-07.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.** "Explicame este código de GlobalDesk 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 GlobalDesk 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.