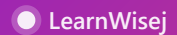


Ejercicio de programación con IA

Construye el ejemplo SupportDesk 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 «Enlace bidireccional y editores CRUD» 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

Construye los formularios modales Add Ticket y Edit Ticket de Support Desk Data Console: un TicketEditModel, un TicketEditorForm con un BindingSource y un ErrorProvider, propiedades de TextBox, ComboBox, DateTimePicker y CheckBox enlazadas al modelo, LoadEditorAsync para tickets nuevos y existentes, un SaveAsync que llame a EndEdit, un marcador de posición para la validación, el mapeo y SaveChangesAsync, Delete con confirmación y una carga nueva de la entidad, y una cuadrícula principal que se actualice tras DialogResult.OK.

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

- ✓ TicketEditModel y TicketEditorForm con un BindingSource y un ErrorProvider
- ✓ Propiedades de TextBox, ComboBox, DateTimePicker y CheckBox enlazadas al modelo de edición
- ✓ LoadEditorAsync para tickets nuevos y existentes y SaveAsync con EndEdit, mapeo y SaveChangesAsync
- ✓ Delete con confirmación, una carga nueva de la entidad y un manejo elegante de los registros ya eliminados
- ✓ Cuadrícula padre actualizada tras DialogResult.OK, y Cancel no persiste nada

Por qué tiene esta forma

Los operadores de Support Desk quieren pantallas Add Ticket y Edit Ticket que se comporten como formularios de escritorio: un cuadro de diálogo modal, botones Save, Cancel y Delete, y una cuadrícula que muestre el cambio en cuanto se cierra el diálogo. El enlace bidireccional hace que esa pantalla se construya rápido, pero Save es una operación deliberada, no un efecto secundario de escribir. Si el editor lee un valor enlazado demasiado pronto, guarda una fila de cuadrícula sin seguimiento como si estuviera rastreada o deja que un doble clic envíe dos veces, el formulario sigue pareciendo correcto mientras pierde o duplica datos en silencio. Este módulo hace explícitos el guardado, la cancelación, la eliminación y

la recarga, para que el editor se sienta como un formulario de escritorio y aun así siga reglas de acceso a datos seguras para la web.

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
enlace bidireccional	Una propiedad de un control, como <code>TextBox.Text</code> , se enlaza a una propiedad del elemento actual, de modo que el control muestra el modelo y el modelo recibe lo que el usuario escribió.
<code>DataSourceUpdateMode</code>	El argumento de <code>DataBindings.Add</code> que decide cuándo un control envía su valor al modelo: al validar (cuando sale el foco) o en cada cambio de propiedad.
<code>EndEdit</code> y <code>CancelEdit</code>	<code>BindingSource.EndEdit</code> confirma en el elemento actual las ediciones pendientes de los controles; <code>CancelEdit</code> las descarta. <code>Save</code> siempre empieza con <code>EndEdit</code> .
<code>AddNew</code> y <code>RemoveCurrent</code>	Operaciones de lista del <code>BindingSource</code> que crean un nuevo elemento actual o quitan el seleccionado de la lista enlazada; ninguna toca la base de datos hasta que guardas.
enlace directo a entidades	Enlazar los controles directamente a una entidad rastreada de EF Core, de modo que el change tracker ya sabe qué ha cambiado. Productivo para formularios pequeños, de vida corta y con Context privado.
modelo de edición	Una clase solo de interfaz, como <code>TicketEditModel</code> , que se enlaza, se valida y después se mapea a la entidad, dando a la validación, el formato y el recorte por seguridad una frontera propia.
guardar y eliminar con Context nuevo	Cada <code>Save</code> y <code>Delete</code> crea su propio <code>DbContext</code> desde la fábrica, carga la entidad por clave, la modifica y llama a <code>SaveChangesAsync</code> , de modo que la fuente de verdad es la base de datos, no la fila de la cuadrícula.

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 4. Puedes abrir tú mismo [module-04.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 SupportDesk 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 4 - «Enlace bidireccional y editores CRUD» - del curso de LearnWisej «Enlace de datos con EF Core», 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/ef-core-binding/specs/module-04.md>

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

1. Proyecto «SupportDesk», 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: Construye los formularios modales Add Ticket y Edit Ticket de Support Desk Data Console: un `TicketEditModel`, un `TicketEditorForm` con un `BindingSource` y un `ErrorProvider`, propiedades de `TextBox`, `ComboBox`, `DateTimePicker` y `CheckBox` enlazadas al modelo, `LoadEditorAsync` para tickets nuevos y existentes, un `SaveAsync` que llame a `EndEdit`, un marcador de posición para la validación, el mapeo y `SaveChangesAsync`, `Delete` con confirmación y una carga nueva de la entidad, y una cuadrícula principal que se actualice tras `DialogResult.OK`.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - `TicketEditModel` y `TicketEditorForm` con un `BindingSource` y un `ErrorProvider`
 - Propiedades de `TextBox`, `ComboBox`, `DateTimePicker` y `CheckBox` enlazadas al modelo de edición
 - `LoadEditorAsync` para tickets nuevos y existentes y `SaveAsync` con `EndEdit`, mapeo y `SaveChangesAsync`
 - `Delete` con confirmación, una carga nueva de la entidad y un manejo elegante de los registros ya eliminados
 - Cuadrícula padre actualizada tras `DialogResult.OK`, y `Cancel` no persiste nada
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 duración elegiste para el `DbContext` del editor y qué tendría que cumplir el formulario para que un `Context` con ámbito de formulario fuera aceptable?
- ¿Qué valores están enlazados a la interfaz, cuáles viven solo en el modelo de edición y cuáles permanecen solo en la base de datos, y por qué `Number` no es editable?
- ¿Qué modos de fallo manejaste explícitamente (doble clic, ticket ya eliminado, rechazo de la base de datos, edición concurrente) y qué ve el operador en cada caso?

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 duración elegiste para el DbContext del editor y qué tendría que cumplir el formulario para que un Context con ámbito de formulario fuera aceptable?
- ✓ ¿Qué valores están enlazados a la interfaz, cuáles viven solo en el modelo de edición y cuáles permanecen solo en la base de datos, y por qué Number no es editable?
- ✓ ¿Qué modos de fallo manejaste explícitamente (doble clic, ticket ya eliminado, rechazo de la base de datos, edición concurrente) y qué ve el operador en cada caso?

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 4 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 4 en GitHub](#)

O clona el conjunto completo con `git clone https://github.com/iceteagroup/LearnWisej-Samples` y abre `Data Binding with EF Core Course/Module 4`.



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.

- ✓ **Cambia una pieza sin mover las juntas.** *"Elige la parte de este proyecto SupportDesk 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 SupportDesk 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."*
- ✓ **Añade el segundo caso.** *"Amplía este proyecto SupportDesk 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."*



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.