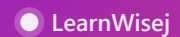


Ejercicio de programación con IA

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 devuelve — 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 SupportDesk; 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

Construye el explorador de tickets de Support Desk: los records TicketListItem y TicketSearchCriteria, un método SearchTicketsAsync que componga AsNoTracking, Where, OrderBy, Skip, Take y Select antes de ToListAsync, un DataGridView enlazado a TicketListItem mediante un BindingSource, ComboBox de búsqueda para estado y cliente cargados antes de la primera búsqueda, los botones Search, Next Page y Previous Page, la visualización del recuento total y del tamaño de página, y un indicador de carga que mantenga la interfaz utilizable después de un error.

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

- ✓ Records TicketListItem y TicketSearchCriteria con SearchTicketsAsync
- ✓ DataGridView enlazado a TicketListItem mediante un BindingSource
- ✓ Filtrado y paginación compuestos en la consulta para que el trabajo lo haga la base de datos
- ✓ ComboBox de consulta cargados antes de la primera búsqueda, que muestran nombres y almacenan claves
- ✓ Botones Search, Next Page y Previous Page con un recuento total y una protección de carga

Por qué tiene esta forma

Los operadores de Support Desk necesitan una cuadrícula de tickets que busque rápido sin traer toda la base de datos a la memoria del servidor. En una aplicación Wisej.NET cada sesión mantiene en el servidor sus propios formularios, BindingSources y filas seleccionadas, así que una pantalla que carga miles de entidades Ticket rastreadas por usuario multiplica ese coste por el número de navegadores abiertos. Este módulo construye bien el explorador de tickets: componer la consulta, ejecutarla de forma asíncrona, materializar una pequeña página proyectada y entregar esa lista a la interfaz mediante un BindingSource.

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
método de ejecución asíncrona	ToListAsync, CountAsync y SingleOrDefaultAsync son las llamadas que realmente envían SQL; esperarlas con await deja libre el hilo del servidor mientras trabaja la base de datos.
proyección	Un Select a un record TicketListItem devuelve exactamente las columnas que muestra la cuadrícula, en lugar de la entidad Ticket con su grafo de Customer, Agent y Comments.
consulta sin seguimiento	AsNoTracking indica al Context que no guarde una instantánea de los resultados: menos memoria, ninguna actualización accidental y una lista independiente de cualquier unidad de trabajo de edición.
BindingSource	El componente de Wisej.NET situado entre una colección de datos y sus controles enlazados; gestiona el elemento actual, la navegación y las operaciones de lista, y mantiene sincronizados la cuadrícula y los controles de detalle.
IQueryable	Una receta para una consulta a la base de datos, no una lista estable: hay que ejecutarla y materializarla antes de asignar nada a la interfaz.
protección de carga	Un indicador, botones desactivados y una etiqueta de estado alrededor de cada llamada a la base de datos esperada con await, restaurados en finally, para que se ejecute una sola operación a la vez y la página siga siendo usable tras un error.
enlace de listas de consulta	Un ComboBox con DisplayMember para el nombre mostrado y ValueMember para la clave almacenada, cargado antes de que se ejecute la primera búsqueda o se asigne el modelo de edición.

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 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 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 3 – «Carga de datos en BindingSource, DataGridView y controles de búsqueda» – 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-03.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 el explorador de tickets de Support Desk: los records `TicketListItem` y `TicketSearchCriteria`, un método `SearchTicketsAsync` que componga `AsNoTracking`, `Where`, `OrderBy`, `Skip`, `Take` y `Select` antes de `ToListAsync`, un `DataGridView` enlazado a `TicketListItem` mediante un `BindingSource`, `ComboBox` de búsqueda para estado y cliente cargados antes de la primera búsqueda, los botones `Search`, `Next Page` y `Previous Page`, la visualización del recuento total y del tamaño de página, y un indicador de carga que mantenga la interfaz utilizable después de un error.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Records `TicketListItem` y `TicketSearchCriteria` con `SearchTicketsAsync`
 - `DataGridView` enlazado a `TicketListItem` mediante un `BindingSource`
 - Filtrado y paginación compuestos en la consulta para que el trabajo lo haga la base de datos
 - `ComboBox` de consulta cargados antes de la primera búsqueda, que muestran nombres y almacenan claves
 - Botones `Search`, `Next Page` y `Previous Page` con un recuento total y una protección de carga
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` dentro de `SearchTicketsAsync` y qué se rompería si la página mantuviera vivo un `Context` durante toda su vida?
- Después de una búsqueda, ¿qué datos se guardan en el servidor para esta sesión y cuáles permanecen solo en la base de datos? ¿Cuánto cuesta un segundo operador simultáneo?
- Si la lista de consulta de clientes creciera hasta cincuenta mil filas, ¿qué partes de la página cambiarías y qué partes del servicio de consulta quedarían exactamente igual?

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 dentro de SearchTicketsAsync y qué se rompería si la página mantuviera vivo un Context durante toda su vida?
- ✓ Después de una búsqueda, ¿qué datos se guardan en el servidor para esta sesión y cuáles permanecen solo en la base de datos? ¿Cuánto cuesta un segundo operador simultáneo?
- ✓ Si la lista de consulta de clientes creciera hasta cincuenta mil filas, ¿qué partes de la página cambiarías y qué partes del servicio de consulta quedarían exactamente igual?

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 `Data Binding with EF Core 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.

- ✓ **Revisalo como un pull request.** *"Explicame este código de SupportDesk 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 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."*



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.