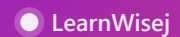


Ejercicio de programación con IA

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.



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

Optimiza el explorador de tickets de Support Desk: activa el logging de EF Core en desarrollo, registra el SQL y los tiempos de la búsqueda sin optimizar, sustituye la carga de entidades por una proyección, añade AsNoTracking a las consultas de la cuadrícula, añade paginación con un tamaño de página de 50, sustituye el acceso diferido a las navegaciones por una proyección o Include, y documenta en las notas del laboratorio los resultados medidos antes y después.

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

- ✓ Registro de EF Core activado en desarrollo, con el SQL y los tiempos sin optimizar anotados
- ✓ Consulta de la cuadrícula reescrita como proyección sin seguimiento
- ✓ Paginación con un tamaño de página de 50 y filtros indexados
- ✓ Ningún comportamiento N+1 de carga diferida en el código de visualización
- ✓ Notas del laboratorio que documenten la mejora medida antes y después

Por qué tiene esta forma

El explorador de tickets de Support Desk funcionaba en el Módulo 3 con cincuenta tickets de ejemplo. Con cincuenta mil no funcionará, a menos que la consulta haga menos trabajo y en el lugar adecuado. El fallo es silencioso: una cuadrícula que tarda tres segundos porque el código de formato tocó una propiedad de navegación por fila, una sesión que retiene miles de entidades con seguimiento que nadie va a editar, una búsqueda que recorre una tabla entera. Este módulo trata de hacer menos trabajo, en SQL, a propósito, y de demostrarlo con un registro en lugar de con una corazonada.

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
proyección	Un Select que devuelve exactamente los campos que necesita la pantalla, como TicketListItem, para que la base de datos combine los nombres relacionados en una sola sentencia.
Include e Include filtrado	Carga anticipada de las entidades relacionadas junto con la raíz; un Include filtrado acota una colección (los últimos cinco comentarios) para que un editor cargue un agregado controlado.
carga explícita	Cargar una navegación más tarde, bajo demanda, mediante Collection o Reference de la entrada y LoadAsync, cuando el usuario despliega una sección de detalle.
carga diferida y N+1	Propiedades de navegación que se cargan de forma transparente en el primer acceso; en una cuadrícula eso supone una consulta oculta por fila o por acceso a navegación.
AsNoTracking y resolución de identidad	Resultados que evitan el change tracker en las pantallas de solo lectura; AsNoTrackingWithIdentityResolution sigue devolviendo una única instancia por entidad cuando la misma fila aparece varias veces.
paginación en la consulta	Where, OrderBy, Skip, Take y Select compuestos antes de ToListAsync, con un CountAsync aparte para el total.
registro de EF Core	El SQL generado de cada comando junto con su duración, mediante LogTo o Microsoft.Extensions.Logging.

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

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 6 – «Cargas asíncronas, datos relacionados, filtrado y rendimiento» – 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-06.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: Optimiza el explorador de tickets de Support Desk: activa el logging de EF Core en desarrollo, registra el SQL y los tiempos de la búsqueda sin optimizar, sustituye la carga de entidades por una proyección, añade `AsNoTracking` a las consultas de la cuadrícula, añade paginación con un tamaño de página de 50, sustituye el acceso diferido a las navegaciones por una proyección o `Include`, y documenta en las notas del laboratorio los resultados medidos antes y después.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Registro de EF Core activado en desarrollo, con el SQL y los tiempos sin optimizar anotados
 - Consulta de la cuadrícula reescrita como proyección sin seguimiento
 - Paginación con un tamaño de página de 50 y filtros indexados
 - Ningún comportamiento N+1 de carga diferida en el código de visualización
 - Notas del laboratorio que documenten la mejora medida antes y después
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:

- Tu registro mostró 151 sentencias para una pantalla de 50 tickets. ¿Qué línea de código provocó las 150 adicionales, y por qué nada en la aplicación te avisó?
- El editor necesita el ticket, su cliente y su categoría, y los últimos cinco comentarios; el panel de comentarios abre el historial completo bajo demanda. ¿Qué estrategia de carga atiende cada una de esas tres necesidades, y por qué la proyección es la respuesta equivocada para el editor?
- Un compañero de equipo propone consultas compiladas y pooling de Contexts para el explorador de tickets antes de mirar el registro. ¿Qué medirías primero, y qué tendrían que mostrar las cifras para que cualquiera de los dos cambios merezca la pena?

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:

- ✓ Tu registro mostró 151 sentencias para una pantalla de 50 tickets. ¿Qué línea de código provocó las 150 adicionales, y por qué nada en la aplicación te avisó?
- ✓ El editor necesita el ticket, su cliente y su categoría, y los últimos cinco comentarios; el panel de comentarios abre el historial completo bajo demanda. ¿Qué estrategia de carga atiende cada una de esas tres necesidades, y por qué la proyección es la respuesta equivocada para el editor?
- ✓ Un compañero de equipo propone consultas compiladas y pooling de Contexts para el explorador de tickets antes de mirar el registro. ¿Qué medirías primero, y qué tendrían que mostrar las cifras para que cualquiera de los dos cambios merezca la pena?

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 6 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 6 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 6`.



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.

- ✓ **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."*
- ✓ **Demuestra que de verdad funciona.** *"Recorre este proyecto entregable por entregable frente a la especificación del módulo en <https://learnwisej.com/assets/courses/ef-core-binding/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 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."*



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.