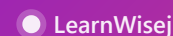


Ejercicio de programación con IA

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.



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 2.

Qué estás pidiendo

Crea las entidades Customer, Agent, Category, Ticket y TicketComment de Support Desk Data Console, configura las relaciones y los comportamientos de eliminación con la Fluent API, configura RowVersion con IsRowVersion, añade índices para Status, DueDate, CustomerId y UpdatedAt, añade una factoría en tiempo de diseño, crea la migración InitialCreate y carga cinco clientes, tres agentes, categorías y al menos cincuenta tickets en una base de datos local SQLite o SQL Server LocalDB.

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

- ✓ Entidades Customer, Agent, Category, Ticket y TicketComment con relaciones y comportamientos de eliminación
- ✓ RowVersion configurado con IsRowVersion
- ✓ Índices para Status, DueDate, CustomerId y UpdatedAt
- ✓ Fábrica en tiempo de diseño y la migración InitialCreate
- ✓ Datos iniciales: cinco clientes, tres agentes, categorías y al menos cincuenta tickets

Por qué tiene esta forma

La Support Desk Data Console mostrará tickets en una cuadrícula paginable, los editará en un formulario modal y permitirá que varios agentes trabajen a la vez sobre la misma cola, y cada pantalla es tan buena como el modelo que tiene debajo. Un filtro sin un índice adecuado se convierte en un recorrido completo de la tabla con cincuenta mil tickets, dos agentes que guardan el mismo ticket pierden en silencio el trabajo de uno de ellos si la fila no lleva un token de concurrencia, y un esquema parcheado a mano rompe el siguiente despliegue. El modelo es el contrato entre la interfaz y la base de datos, así que este módulo lo construye de forma explícita y lo versiona como código fuente.

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
entidad apta para el enlace	Una entidad con una clave predecible, campos de visualización estables, propiedades de navegación explícitas y tablas de consulta pequeñas, de modo que una cuadrícula, un ComboBox y un editor puedan enlazarse a ella.
view model TicketListItem	Una proyección plana con solo las columnas que muestra la cuadrícula, para que la cuadrícula nunca reciba el grafo completo de Ticket y permanezca independiente de la unidad de trabajo de edición.
configuración con Fluent API	Reglas de mapeo escritas en OnModelCreating (cadenas obligatorias, longitudes, índices, comportamiento de eliminación, precisión, valores predeterminados) para que las reglas de persistencia vivan junto al DbContext.
token de concurrency RowVersion	Un array de bytes generado por la base de datos y marcado con IsRowVersion; EF Core añade su valor original a la cláusula WHERE del UPDATE, de modo que un guardado obsoleto actualiza cero filas y lanza DbUpdateConcurrencyException.
comportamiento de eliminación	La regla OnDelete de una relación (Restrict, SetNull, Cascade) que decide si eliminar un cliente se rechaza, desasigna sus tickets o elimina también los comentarios.
migración	Una clase C# más la instantánea del modelo que describe un cambio de esquema revisado, confirmada en el control de código fuente y aplicada en orden para llevar cualquier base de datos al estado del modelo.
fábrica en tiempo de diseño	Una IDesignTimeDbContextFactory que indica a las herramientas dotnet ef cómo construir el Context sin ejecutar el arranque de la aplicación Wisej.NET.

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 2. Puedes abrir tú mismo [module-02.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 2 – «Modelado, configuración del DbContext y migraciones» – 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-02.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: Crea las entidades Customer, Agent, Category, Ticket y TicketComment de Support Desk Data Console, configura las relaciones y los comportamientos de eliminación con la Fluent API, configura RowVersion con IsRowVersion, añade índices para Status, DueDate, CustomerId y UpdatedAt, añade una factoría en tiempo de diseño, crea la migración InitialCreate y carga cinco clientes, tres agentes, categorías y al menos cincuenta tickets en una base de datos local SQLite o SQL Server LocalDB.
3. Genera estos entregables e indícame dónde está cada uno dentro de la solución:
 - Entidades Customer, Agent, Category, Ticket y TicketComment con relaciones y comportamientos de eliminación
 - RowVersion configurado con IsRowVersion
 - Índices para Status, DueDate, CustomerId y UpdatedAt
 - Fábrica en tiempo de diseño y la migración InitialCreate
 - Datos iniciales: cinco clientes, tres agentes, categorías y al menos cincuenta tickets
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:

- Eliminas un cliente que todavía tiene tickets abiertos. ¿Qué ocurre con el comportamiento de eliminación que configuraste y qué habría ocurrido con Cascade?
- Dos agentes cargan el ticket SD-1042 y ambos hacen clic en Save con un segundo de diferencia. ¿Qué guardado lanza la excepción, qué contiene la cláusula WHERE de la sentencia UPDATE y qué ve el usuario?
- ¿Por qué IsRequired() sobre Title no le da al usuario un mensaje «Enter a title», y de dónde saldrá ese mensaje más adelante en el curso?

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:

- ✓ Eliminas un cliente que todavía tiene tickets abiertos. ¿Qué ocurre con el comportamiento de eliminación que configuraste y qué habría ocurrido con Cascade?
- ✓ Dos agentes cargan el ticket SD-1042 y ambos hacen clic en Save con un segundo de diferencia. ¿Qué guardado lanza la excepción, qué contiene la cláusula WHERE de la sentencia UPDATE y qué ve el usuario?
- ✓ ¿Por qué IsRequired() sobre Title no le da al usuario un mensaje «Enter a title», y de dónde saldrá ese mensaje más adelante en el curso?

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 2 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 2 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 2`.



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.

- ✓ **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-02.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."*
- ✓ **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."*



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.