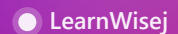


Qué es Wisej.NET (y cuándo usarlo)

Una imagen clara del framework antes de escribir una sola línea de código.



Objetivo de la lección

Al terminar esta lectura podrás explicar qué es Wisej.NET, describir su modelo mental y decidir cuándo es la herramienta adecuada para un proyecto.

1. Qué es Wisej.NET

Wisej.NET es un framework y un entorno de desarrollo para crear **aplicaciones web en tiempo real** con los lenguajes .NET y el flujo de trabajo de Visual Studio que muchos desarrolladores de escritorio ya conocen. Una aplicación Wisej.NET se ejecuta en un navegador, pero la lógica de la aplicación se escribe principalmente **en C# o VB.NET en el servidor**.

Un primer modelo mental útil: **con Wisej.NET parece que construyes una aplicación de estilo escritorio, pero la aplicación terminada se entrega a través del navegador web**. En lugar de escribir a mano cada fragmento de HTML, CSS y JavaScript, trabajas con controles, propiedades y eventos de Wisej.NET y con código del lado del servidor.



Definición en pocas palabras

Wisej.NET permite a un desarrollador .NET crear aplicaciones web con herramientas de diseño visual, controles y código C# basado en eventos, en lugar de partir de un framework JavaScript de frontend.

2. Cómo funciona

En una aplicación Wisej.NET típica, el navegador muestra la interfaz de usuario mientras el servidor gestiona los objetos de la aplicación y la lógica de los eventos. Cuando un usuario hace clic en un botón, edita un campo o selecciona una fila, el evento lo gestiona código C# en el servidor. Wisej.NET mantiene sincronizados el navegador y el servidor para que la interfaz se actualice sin interrupciones.

Parte	Qué significa para un principiante
Navegador	Donde el usuario ve la aplicación e interactúa con ella: páginas, botones, etiquetas, cuadrículas y formularios.
Servidor	Ejecuta el código de la aplicación .NET. Aquí se gestionan los clics en los botones y la mayoría de los eventos.
Designer	La superficie visual de Visual Studio donde organizas los controles y estableces sus propiedades.
Controles	Bloques de interfaz reutilizables: <code>Button</code> , <code>Label</code> , <code>TextBox</code> , <code>DataGridView</code> , <code>Panel</code> , <code>Page</code> , <code>Window</code> .
Eventos	Cosas que ocurren (<code>Click</code> , <code>TextChanged</code> , <code>SelectedIndexChanged</code>) y que ejecutan tu código.

3. Cuándo usar Wisej.NET

Wisej.NET destaca cuando el proyecto es una **aplicación de negocio** y el equipo quiere un desarrollo rápido con .NET, Visual Studio y un diseñador visual, sobre todo cuando la aplicación tiene muchos formularios, muchos datos o un estilo de escritorio.

Encaja bien	Por qué ayuda Wisej.NET
Herramientas internas de negocio	Dashboards, formularios, pantallas de administración y herramientas de flujo de trabajo, construidos rápidamente.
Aplicaciones CRUD	Los controles se corresponden de forma natural con crear / leer / actualizar / eliminar registros.
Aplicaciones con mucho DataGridView	Las tablas de datos, las listas de registros y las vistas de detalle son patrones centrales de Wisej.NET.
Modernización de aplicaciones de escritorio	La experiencia de estilo WinForms se mantiene al pasar a la web.
Prototipos rápidos	El Designer y el modelo de eventos producen pantallas funcionales muy deprisa.

4. Cuándo *no* usarlo

- Usa un sitio web tradicional o un CMS para páginas de contenido principalmente públicas y orientadas al SEO.
- Usa un framework de frontend cuando necesites específicamente una arquitectura de componentes JavaScript personalizada.
- Usa un generador de páginas estáticas cuando no haya un flujo de negocio real ni interacción con una base de datos.
- Usa el desarrollo móvil nativo cuando el objetivo principal sea una integración profunda con el dispositivo y la distribución en tiendas de aplicaciones.

5. El modelo mental de Wisej.NET

Piensa en términos de **pantallas, controles, propiedades y eventos**. Una pantalla contiene controles. Los controles tienen nombres y propiedades. Los usuarios interactúan con los controles. Los eventos ejecutan código, y ese código puede actualizar la interfaz, llamar a un servicio, guardar datos o navegar.

Flujo de ejemplo

```
El usuario hace clic en
→ se ejecuta el evento btnSave_Click
→ lee los valores de los cuadros de texto
→ llama a un método de servicio
→ el servicio guarda los datos
→ la interfaz muestra un mensaje de estado
```



Hábito de principiante

No pongas toda la lógica de negocio dentro del clic del botón. Mantén el clic legible y llama a un método auxiliar, a un servicio o a una clase repositorio cuando la lógica crezca.

Vocabulario clave

- **Proyecto:** la aplicación completa en Visual Studio: código, archivos del Designer, referencias y configuración.
- **Page / Window:** una pantalla o vista que se muestra al usuario (las Windows suelen tener estilo de diálogo).
- **Control:** un objeto de la interfaz, como un botón, una etiqueta, un cuadro de texto, una cuadrícula o una casilla.
- **Propiedad:** un ajuste de un control: `Text` , `Name` , `Dock` , `Size` , `Visible` .
- **Evento:** algo que ocurre (un clic, un cambio de selección) y que ejecuta código.
- **Code-behind:** el archivo `.cs` normal donde vive el comportamiento.
- **Clase de servicio:** una clase separada que contiene la lógica de negocio y el acceso a datos, lejos de la interfaz.