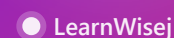


De WinForms a la web — Libro de lecciones

Todas las lecciones de los módulos de este curso, reunidas en un solo documento.



Índice

01	Análisis de la migración y mapa de riesgos	Módulo 1 · Guía de la lección
02	Análisis de la migración y mapa de riesgos	Módulo 1 · Guía del laboratorio / examen
03	Conversión del proyecto y arranque de Wisej.NET	Módulo 2 · Guía de la lección
04	Conversión del proyecto y arranque de Wisej.NET	Módulo 2 · Guía del laboratorio / examen
05	Formularios, navegación, diseños y flujo modal	Módulo 3 · Guía de la lección
06	Formularios, navegación, diseños y flujo modal	Módulo 3 · Guía del laboratorio / examen
07	Sesiones, miembros estáticos, contexto de usuario y seguridad multiusuario	Módulo 4 · Guía de la lección
08	Sesiones, miembros estáticos, contexto de usuario y seguridad multiusuario	Módulo 4 · Guía del laboratorio / examen
09	Acceso a datos, DataGridView, validación y rendimiento	Módulo 5 · Guía de la lección
10	Acceso a datos, DataGridView, validación y rendimiento	Módulo 5 · Guía del laboratorio / examen
11	Archivos, informes, automatización de Office y límites del navegador	Módulo 6 · Guía de la lección
12	Archivos, informes, automatización de Office y límites del navegador	Módulo 6 · Guía del laboratorio / examen
13	Modernización, interfaz adaptable, seguridad, despliegue y proyecto final	Módulo 7 · Guía de la lección
14	Modernización, interfaz adaptable, seguridad, despliegue y proyecto final	Módulo 7 · Guía del laboratorio / examen

Análisis de la migración y mapa de riesgos

Evalúa una aplicación WinForms y elabora un backlog de migración realista antes de modificar el código. Por qué importa, la lectura principal y los objetivos de aprendizaje: léelo antes del laboratorio.



Enfoque de la migración

Evalúa una aplicación WinForms y elabora un backlog de migración realista antes de modificar el código.

Por qué importa este módulo

La migración de escritorio a web cambia el entorno de ejecución mucho más que el lenguaje C#. Los estudiantes aprenden a separar la lógica de negocio portable de las suposiciones de escritorio, a identificar las dependencias de riesgo y a elegir una secuencia de migración.

Lectura principal

Una migración de WinForms a la web con éxito empieza antes de tocar el código. Los sistemas de escritorio heredados suelen guardar años de decisiones de negocio en manejadores de eventos, código de validación, informes, cuadrículas y pequeñas clases auxiliares. Reescribir desde cero puede parecer más limpio al principio, pero normalmente convierte un comportamiento de negocio conocido en un largo proyecto de redescubrimiento. Una migración a Wisej.NET ofrece al equipo otro camino: conservar el modelo C# orientado a eventos y llevar la interfaz de usuario a un runtime nativo del navegador.

La fase de evaluación tiene dos entregables: un inventario de la aplicación y un backlog de migración. El inventario enumera pantallas, controles, dependencias de terceros, informes, recursos, dependencias de base de datos, archivos locales, claves del registro, llamadas de automatización de Office, llamadas al shell y a procesos, impresoras, trabajos en segundo plano y estado global. El backlog convierte el inventario en elementos de trabajo, cada uno con una categoría de esfuerzo y una categoría de riesgo.

Usa cuatro categorías para cada pantalla o subsistema. Los elementos de port directo usan sobre todo controles WinForms estándar y manejadores de eventos corrientes. Los elementos de port con adaptación son compatibles, pero contienen suposiciones de escritorio como rutas de archivos locales, acceso al registro, impresión o estado de usuario estático. Los elementos de rediseño dependen de un componente exclusivo de escritorio o están tan recargados que conviene un diseño pensado primero para el navegador. Los elementos aplazados son rutas poco frecuentes, herramientas solo para administradores o funciones que pueden quedar fuera de la primera versión web.

El primer corte migrado debe ser deliberadamente pequeño, pero representativo. Debe incluir el arranque de la aplicación, el contexto de usuario/sesión, la navegación, una cuadrícula enlazada a datos, un editor modal, un informe o una exportación y una operación con archivos. Este corte saca a la luz pronto los problemas difíciles y evita la falsa sensación de avance que produce portar solo los formularios fáciles.

Plantilla de evaluación de la migración

- ✓ Área
- ✓ Preguntas que responder

- ✓ Decisión
- ✓ Formularios y controles
- ✓ ¿Cuántas pantallas? ¿Controles estándar? ¿Controles personalizados? ¿Controles de terceros?
- ✓ Port directo / adaptar / rediseñar.
- ✓ Lógica de negocio
- ✓ ¿La lógica está mezclada con los manejadores de eventos de la interfaz? ¿Se puede extraer o reutilizar tal cual?
- ✓ Reutilizar / extraer / probar.
- ✓ Acceso a datos
- ✓ ¿ADO.NET, EF, datasets tipados, SQL directo en los eventos, consultas de larga duración?
- ✓ Portar / optimizar / capa de servicios.
- ✓ Estado
- ✓ ¿Miembros estáticos, servicios singleton, usuario actual, empresa actual, conexión actual?
- ✓ Caché global / sesión / base de datos.
- ✓ Límite de escritorio
- ✓ ¿Archivos, registro, Office COM, impresoras, comandos de shell, API de Windows?
- ✓ Sustituir por un patrón seguro para la web.
- ✓ Despliegue
- ✓ ¿IIS, Docker, Linux, dependencias exclusivas de Windows, licencias, escalado?
- ✓ Entorno de destino y restricciones.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Inventariar formularios, controles, componentes de terceros, recursos, acceso a datos, informes, automatización de Office, uso del registro, uso del sistema de archivos y estado estático.
- ✓ Clasificar cada pantalla como port directo, port con adaptación, rediseño o aplazamiento.
- ✓ Elegir un corte vertical que valide pronto el arranque, el inicio de sesión, la navegación, los datos, el flujo modal, los archivos y los informes.
- ✓ Documentar criterios de aceptación innegociables: primero la paridad, después la modernización y, por último, el despliegue.



Claves de la migración

- ✓ Una migración de WinForms a Wisej.NET debe empezar con un inventario y un backlog basado en el riesgo, no sustituyendo espacios de nombres al azar.
- ✓ El primer corte vertical debe validar el arranque, la navegación, los datos, el flujo modal, el comportamiento de archivos e informes y el contexto de sesión.
- ✓ Las decisiones de migración deben clasificarse como port directo, adaptación, rediseño, aplazamiento o eliminación.

Análisis de la migración y mapa de riesgos

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Crea un libro de evaluación de la migración para la aplicación WinForms que estás migrando (LegacyOrderDesk en los ejemplos del curso) y ordena sus pantallas por esfuerzo y riesgo.

Pasos del laboratorio

- ✓ Abre la aplicación WinForms que estás migrando, o cualquier aplicación WinForms pequeña con inicio de sesión, una pantalla de lista y un informe.
- ✓ Crea un inventario de formularios, controles, informes, recursos, orígenes de datos, archivos, claves del registro y campos estáticos.
- ✓ Etiqueta cada elemento como port directo, adaptación, rediseño, aplazamiento o eliminación.
- ✓ Elige el primer corte vertical y redacta sus criterios de aceptación.
- ✓ Haz commit del libro de evaluación en la rama de migración.

Conversión del proyecto y arranque de Wisej.NET

Convierte o recrea el shell del proyecto para que un formulario migrado pueda ejecutarse en el navegador. Por qué importa, la lectura principal y los objetivos de aprendizaje: léelo antes del laboratorio.



Enfoque de la migración

Convierte o recrea el shell del proyecto para que un formulario migrado pueda ejecutarse en el navegador.

Por qué importa este módulo

El port inicial establece el modelo de alojamiento ASP.NET/ASP.NET Core, las referencias a los paquetes de Wisej.NET, los archivos de arranque predeterminados, el comportamiento de los recursos y el punto de entrada que sustituye al patrón Application.Run de WinForms.

Lectura principal

El primer hito técnico es un proyecto Wisej.NET capaz de alojar la aplicación migrada. Hay dos caminos habituales. Uno empieza con un proyecto Wisej.NET nuevo y copia en él formularios, recursos, clases y referencias. El otro modifica el archivo de proyecto existente para convertirlo en una aplicación web. El camino más seguro para la formación es crear una rama, crear un proyecto Wisej.NET nuevo e ir trayendo los formularios de uno en uno hasta entender los patrones.

El arranque de Wisej.NET incluye archivos orientados al navegador y un punto de entrada en el servidor. Default.html carga el script del cliente y el shell de la página. Default.json indica a Wisej.NET qué método del servidor o qué vista principal debe iniciar. Program.cs contiene el método de arranque del servidor y configura Wisej.NET en la canalización de ASP.NET. Un Program.Main de WinForms convertido no puede llamar a Application.EnableVisualStyles ni a Application.Run de la misma forma; en su lugar, muestra el primer formulario o la primera página de Wisej.NET desde el método de arranque configurado.

El mayor cambio mecánico es la sustitución de espacios de nombres. Las referencias estándar de WinForms como System.Windows.Forms.Button pasan a ser Wisej.Web.Button. Esto resulta familiar a propósito, pero no todas las propiedades o métodos tienen una correspondencia exacta. El flujo de migración consiste en sustituir los espacios de nombres, compilar, clasificar los errores del compilador, corregir las diferencias evidentes y anotar todo lo que no lo sea. No edites los archivos del diseñador a ciegas sin control de código fuente ni copias de seguridad.

Nota sobre versiones

La documentación de migración de Wisej.NET muestra históricamente ejemplos con Wisej-3 y frameworks de destino mixtos, como net48 más un destino .NET. Para un proyecto nuevo del curso, usa el estándar actual de tu organización. Para Wisej.NET 4, el paquete NuGet es Wisej-4 y los proyectos .NET actuales deben seguir las indicaciones de Wisej.NET 4, incluidos los cambios en el administrador de recursos cuando corresponda.

Ejemplo mínimo de arranque

```
using Wisej.Web;
namespace LegacyOrderDesk.Web
{
    internal static class Program
    {
        public static void Main()
        {
            Application.MainPage = new MainPage();
        }
    }
}
```

Patrón de sustitución de espacios de nombres

```
// Before
using System.Windows.Forms;

// After
using Wisej.Web;

// Typical class declaration remains familiar:
public partial class CustomerForm : Form
{
    public CustomerForm()
    {
        InitializeComponent();
    }
}
```



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Crear un proyecto Wisej.NET nuevo o convertir el archivo de proyecto original usando una rama controlada.
- ✓ Entender Default.html, Default.json, Program.cs, los métodos de arranque y las opciones de arranque mainWindow.
- ✓ Sustituir las referencias a System.Windows.Forms por Wisej.Web y resolver los errores del compilador de forma sistemática.
- ✓ Adaptar las cadenas de conexión de app.config, los recursos incrustados, los archivos generados por el diseñador y las referencias a paquetes.



Claves de la migración

- ✓ El arranque de Wisej.NET usa archivos orientados al navegador y una configuración de arranque en el servidor, en lugar del modelo clásico de arranque del ejecutable WinForms.
- ✓ Los controles WinForms estándar suelen pasar de System.Windows.Forms a Wisej.Web, pero sus propiedades y métodos siguen necesitando una revisión guiada por el compilador.
- ✓ Las cadenas de conexión y la configuración de app.config deben adaptarse a la configuración del proyecto web Wisej.NET.

Conversión del proyecto y arranque de Wisej.NET

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Migra el primer formulario a un shell de aplicación Wisej.NET, ejecútalo y documenta cada categoría de error del compilador.

Pasos del laboratorio

- ✓ Crea una rama de migración.
- ✓ Crea un proyecto Wisej.NET nuevo o convierte el shell según el estándar del equipo.
- ✓ Copia el primer formulario y sus dependencias.
- ✓ Sustituye System.Windows.Forms por Wisej.Web.
- ✓ Compila, clasifica los errores del compilador, corrige las diferencias directas y ejecuta la aplicación.
- ✓ Documenta las diferencias restantes no admitidas o poco claras.

Formularios, navegación, diseños y flujo modal

Conserva el flujo de trabajo del usuario de WinForms mientras adaptas los patrones de interfaz de escritorio al comportamiento del navegador. Por qué importa, la lectura principal y los objetivos de aprendizaje: léelo antes del laboratorio.



Enfoque de la migración

Conserva el flujo de trabajo del usuario de WinForms mientras adaptas los patrones de interfaz de escritorio al comportamiento del navegador.

Por qué importa este módulo

Una aplicación Wisej.NET migrada puede conservar formularios, manejadores de eventos, menús, conceptos MDI y diálogos modales familiares. El trabajo consiste en eliminar las suposiciones exclusivas de escritorio y usar las capacidades de diseño y del diseñador de Wisej.NET allí donde el navegador impone nuevas restricciones.

Lectura principal

Los desarrolladores WinForms esperan que las pantallas tengan formularios, controles, eventos, diálogos modales, menús, botones de barra de herramientas y un orden de foco predecible. Wisej.NET conserva buena parte de este modelo. El navegador solo cambia el entorno de despliegue y de renderizado, no el hecho de que los manejadores de eventos C# se ejecutan en el servidor. Esa es la razón principal por la que la migración puede conservar tanto código existente.

Convertir el diseño de las pantallas es un equilibrio entre paridad y mejora. Si un formulario ya usa bien los anclajes y el acoplamiento, puede portarse sin problemas. Si se diseñó para una resolución fija de escritorio de 1024x768, haz primero que funcione y modernízalo después con contenedores, acoplamiento, propiedades adaptables y perfiles de cliente. No dejes que la perfección del diseño bloquee al principio el primer corte funcional.

El flujo modal es una gran ventaja para las aplicaciones de negocio, porque el código puede mostrar un diálogo y continuar cuando el usuario lo cierra. La trampa de la migración es el ciclo de vida de los objetos. El código de escritorio suele confiar en que los formularios se liberen automáticamente o al terminar el proceso. En la aplicación web, los diálogos cerrados que no se reutilizan deben liberarse de forma explícita. La regla más sencilla es: crea los diálogos transitorios en una instrucción `using` o llama a `Dispose` cuando `ShowDialog` devuelva el control.

```
private void editButton_Click(object sender, EventArgs e)
{
    var customer = GetSelectedCustomer();
    using (var dialog = new CustomerEditDialog(customer))
    {
        if (dialog.ShowDialog() == DialogResult.OK)
        {
            SaveCustomer(dialog.Customer);
            MessageBox.Show("Customer saved.");
            ReloadGrid();
        }
    }
}
```

No todos los MessageBox deben seguir siendo un MessageBox modal. Las confirmaciones, las validaciones y las operaciones peligrosas pueden seguir necesitando diálogos bloqueantes. Los mensajes informativos del tipo «guardado correctamente» son mejores candidatos para MessageBox o Toast, porque ahorran clics y resultan más modernos.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Asignar los patrones Form, UserControl, menú, barra de herramientas, pestañas y MDI de WinForms a sus equivalentes en Wisej.NET.
- ✓ Usar el acoplamiento, el anclaje, los contenedores de diseño, el esquema del documento, el orden de tabulación y los perfiles del diseñador para conservar la usabilidad.
- ✓ Usar los diálogos modales de forma segura y liberar los diálogos cerrados cuando no se reutilizan.
- ✓ Sustituir los MessageBox informativos por MessageBox o Toast cuando no se necesita una respuesta bloqueante.



Claves de la migración

- ✓ Wisej.NET admite el flujo modal, pero los diálogos transitorios deben liberarse al cerrarse si no se reutilizan.
- ✓ Los MessageBox informativos son candidatos para MessageBox o Toast; los diálogos que piden una decisión pueden seguir siendo modales.
- ✓ El acoplamiento, el anclaje, los contenedores, el esquema del documento y la vista previa de perfiles de cliente ayudan a conservar y mejorar los diseños WinForms.

Formularios, navegación, diseños y flujo modal

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Porta la navegación y tres pantallas y, después, sustituye un MessageBox innecesario por una notificación no bloqueante.

Pasos del laboratorio

- ✓ Porta el shell principal y la navegación.
- ✓ Porta un diálogo modal de edición.
- ✓ Verifica los valores de retorno del diálogo y su liberación con Dispose.
- ✓ Ajusta el acoplamiento (docking), el anclaje y el orden de tabulación.
- ✓ Sustituye un MessageBox informativo por MessageBox o Toast.
- ✓ Guarda capturas de pantalla del antes y el después.

Sesiones, miembros estáticos, contexto de usuario y seguridad multiusuario

Haz que la aplicación migrada sea segura para muchas sesiones de navegador simultáneas. Por qué importa, la lectura principal y los objetivos de aprendizaje: léelo antes del laboratorio.



Enfoque de la migración

Haz que la aplicación migrada sea segura para muchas sesiones de navegador simultáneas.

Por qué importa este módulo

La mayoría de las aplicaciones WinForms se escribieron con un proceso por usuario. Una aplicación web atiende a muchos usuarios en un único proceso del servidor. Los campos estáticos, los objetos de usuario en caché, la configuración del registro, los archivos abiertos y los indicadores globales deben revisarse y trasladarse a la sesión, a la base de datos o a servicios compartidos seguros.

Lectura principal

Una aplicación WinForms suele ejecutar un proceso por usuario. Por eso las variables estáticas parecen a menudo seguras: solo hay un usuario dentro de ese proceso. En una aplicación web, muchas sesiones pueden ejecutarse dentro del mismo proceso del servidor. Un `CurrentUser`, `CurrentCompany`, `CurrentConnection` o `CurrentOrder` estático pasa a ser compartido si no se refactoriza. Es una de las revisiones más importantes de la migración.

No todos los miembros estáticos son un error. Los datos de consulta estáticos, como listas de países, símbolos de moneda o configuración inmutable, pueden seguir siendo globales si son seguros para subprocesos e independientes de un usuario concreto. El estado por usuario debe trasladarse a una sesión, a un servicio de contexto de usuario, a una fila de base de datos o a otro almacenamiento con ámbito. La pregunta clave es: ¿este valor sería distinto si dos usuarios abrieran la aplicación a la vez? Si la respuesta es sí, no es un estático global seguro.

Patrón de contexto de sesión

```
public sealed class UserSessionContext
{
    public int UserId { get; set; }
    public string UserName { get; set; }
    public int CompanyId { get; set; }
    public string Culture { get; set; }
}

public static class SessionContext
{
    public static UserSessionContext Current
    {
        get
        {
            if (Application.Session.UserContext == null)
                Application.Session.UserContext = new UserSessionContext();
            return Application.Session.UserContext;
        }
    }
}
```

El acceso al registro exige una decisión parecida. En una aplicación web, el código del servidor accede al registro del servidor, no al registro del escritorio del usuario. Las preferencias del usuario deberían convertirse normalmente en configuración respaldada por base de datos, archivos de perfil en el servidor o almacenamiento del navegador. El almacenamiento del navegador es útil para preferencias de interfaz propias de cada cliente, pero la configuración de negocio importante debe estar en manos del servidor.

Hay que planificar el tiempo de espera y la limpieza de la sesión. Una pestaña del navegador puede quedar abandonada, un dispositivo puede entrar en suspensión y la red puede caerse. Las transacciones de larga duración, los archivos temporales, los trabajos de informes y los bloqueos deben liberarse cuando el usuario completa el flujo o durante la limpieza de la sesión.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Distinguir los estáticos globales seguros del estado por usuario/sesión.
- ✓ Sustituir la configuración del registro de escritorio por filas de base de datos, configuración de usuario en el servidor o almacenamiento del navegador según convenga.
- ✓ Entender el ciclo de vida de la sesión, keep-alive, el tiempo de espera, Application.Session y los hooks de limpieza.
- ✓ Crear un servicio de contexto de usuario y un patrón de auditoría del estado estático para el código migrado.



Claves de la migración

- ✓ Los campos estáticos que contienen estado por usuario no son seguros en una aplicación web multiusuario.
- ✓ Los datos de consulta globales inmutables pueden seguir siendo estáticos si son seguros para subprocesos y no dependen del usuario.
- ✓ En el código del servidor, el acceso al registro apunta al registro del servidor, así que la configuración de usuario necesita sustituirse por base de datos, perfil en el servidor o almacenamiento del navegador.

Sesiones, miembros estáticos, contexto de usuario y seguridad multiusuario

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Localiza el estado de usuario estático compartido en tu aplicación WinForms y refactorízalo en un servicio de contexto consciente de la sesión.

Pasos del laboratorio

- ✓ Busca campos estáticos y estado en singletons.
- ✓ Clasifica cada miembro estático como dato compartido inmutable o como estado por usuario.
- ✓ Crea `UserSessionContext`.
- ✓ Traslada al contexto de sesión los valores de usuario actual, empresa actual y filtro actual.
- ✓ Prueba dos sesiones de navegador con usuarios distintos.
- ✓ Documenta los requisitos de limpieza para el tiempo de espera de la sesión y el cierre de sesión.

Acceso a datos, DataGridView, validación y rendimiento

Porta las partes con mucha carga de datos de una aplicación WinForms sin crear pantallas de navegador lentas o con demasiado tráfico. Por qué importa, la lectura principal y los objetivos de aprendizaje: léelo antes del laboratorio.



Enfoque de la migración

Porta las partes con mucha carga de datos de una aplicación WinForms sin crear pantallas de navegador lentas o con demasiado tráfico.

Por qué importa este módulo

Las pantallas WinForms enlazadas a datos suelen portarse bien, pero el navegador cambia la latencia percibida, el volumen de datos y los patrones de interacción. Wisej.NET admite el enlace de datos al estilo WinForms, potentes capacidades de DataGridView, filas virtuales, eventos del servidor y actualizaciones en tiempo real; los estudiantes aprenden cuándo conservar y cuándo rediseñar.

Lectura principal

La mayor parte del código de acceso a datos puede reutilizarse si no presupone una duración de conexión propia del escritorio ni bloquea la interfaz con operaciones grandes. El navegador añade latencia de red y problemas de escalado, pero Wisej.NET sigue permitiendo que el C# del servidor gestione los eventos y actualice los controles. El objetivo es reducir la transferencia de datos innecesaria y evitar cargar conjuntos de datos enormes en los controles solo porque la antigua aplicación de escritorio lo hacía.

DataGridView suele ser el control más importante de una migración. El DataGridView de Wisej.NET admite el enlace de datos habitual y añade funciones a escala web como filas virtuales, controles alojados en la cuadrícula, botones de herramientas, editores de celda personalizados, contenido HTML, dibujo personalizado, filas inmovilizadas, filas y columnas combinadas y filas de resumen. Usa estas funciones para conservar pantallas de negocio densas mejorando a la vez el rendimiento.

Una cuadrícula portada debe probarse con un número de filas similar al de producción. Una cuadrícula que funciona con 100 filas en un laboratorio puede no funcionar de forma aceptable con 500.000 filas. Usa filtrado en el servidor, carga diferida, filas virtuales o vistas de resumen. Envía solo lo que el usuario puede razonablemente examinar o sobre lo que puede actuar.

Pensar en filas virtuales

No enlaces todas las filas si el usuario solo ve un área visible del tamaño de una página.

Mantén el modelo de filtrado y ordenación en el servidor cuando el conjunto de datos sea grande.

Almacena en caché los datos de consulta inmutables, pero no guardes en una caché global los permisos de fila específicos de cada usuario.

Mide el tamaño de la carga útil y la frecuencia de eventos con las herramientas de desarrollo del navegador.

Trata las actualizaciones en tiempo real como una función de negocio, no como un comportamiento predeterminado de todas las cuadrículas.

Patrón de validación

```
private bool ValidateOrder(Order order)
{
    errorProvider1.Clear();
    if (order.CustomerId == 0)
    {
        errorProvider1.SetError(customerComboBox, "Select a customer.");
        return false;
    }
    if (order.Lines.Count == 0)
    {
        MessageBox.Show("Add at least one order line.");
        return false;
    }
    return true;
}
```



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Portar el enlace de datos manteniendo las reglas de negocio y la validación en el servidor.
- ✓ Usar filas virtuales de DataGridView o patrones similares a la paginación para conjuntos de datos grandes.
- ✓ Evitar cargar datos innecesarios en los controles solo porque la aplicación de escritorio lo hacía.
- ✓ Usar WebSocket/server push solo cuando la pantalla se beneficie de verdad de las actualizaciones en tiempo real.



Claves de la migración

- ✓ Las pantallas con DataGridView deben probarse con un número de filas realista y optimizarse con filtrado en el servidor, carga diferida o filas virtuales cuando haga falta.
- ✓ La validación debe permanecer en el código del servidor o de negocio y mostrar una respuesta clara en la interfaz.
- ✓ Las actualizaciones en tiempo real deben añadirse cuando sean útiles, no aplicarse indiscriminadamente a todos los controles.

Acceso a datos, DataGridView, validación y rendimiento

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Porta una cuadrícula de clientes/pedidos, añade validación y sustituye la carga de la tabla completa por un patrón incremental o virtualizado.

Pasos del laboratorio

- ✓ Porta la cuadrícula de clientes/pedidos.
- ✓ Añade validación al flujo de edición.
- ✓ Prueba con un número elevado de filas.
- ✓ Sustituye la carga de la tabla completa por un filtro en el servidor o un acceso virtualizado.
- ✓ Mide la carga inicial y el tiempo de las interacciones habituales.
- ✓ Anota las observaciones de rendimiento en el registro de migración.

Archivos, informes, automatización de Office y límites del navegador

Sustituye las suposiciones de escritorio sobre archivos, registro, informes y Office por flujos seguros para el navegador y el servidor. Por qué importa, la lectura principal y los objetivos de aprendizaje: léelo antes del laboratorio.



Enfoque de la migración

Sustituye las suposiciones de escritorio sobre archivos, registro, informes y Office por flujos seguros para el navegador y el servidor.

Por qué importa este módulo

El navegador no es el escritorio. El código que lee C:\Temp, escribe en HKCU, automatiza Excel, abre archivos locales o espera una impresora local pasa a ser código del servidor, salvo que haya una mediación explícita. Los estudiantes sustituyen estos patrones por carga/descarga, almacenamiento en el servidor, renderizado de PDF, bibliotecas de documentos seguras para el servidor, colas y ClientFileSystem cuando corresponda.

Lectura principal

El navegador no es el escritorio del usuario. Cuando el código migrado del servidor llama a `File.ReadAllText`, `Directory.GetFiles`, `Registry.CurrentUser`, `Process.Start` o `Excel.Application`, ese código se ejecuta en el entorno del servidor, salvo que se use explícitamente una función mediada por el cliente. Estas áreas exigen un rediseño deliberado.

El manejo de archivos tiene varios patrones. La carga lleva un archivo del cliente al servidor para procesarlo. La descarga entrega al usuario un archivo generado en el servidor. El almacenamiento en el servidor guarda los documentos en una ubicación controlada o en una base de datos. La interacción al estilo ClientFileSystem puede usarse cuando la aplicación necesita trabajar más directamente con archivos del cliente y la plataforma lo permite. Los despliegues en Linux también exigen atención a las rutas que distinguen mayúsculas de minúsculas y a las barras normales.

Normalmente, los informes deben generarse en el servidor como PDF y mostrarse con PDFViewer u ofrecerse como descarga. Si el motor de informes no soporta un uso paralelo intenso, introduce una cola. Una cola también mejora la experiencia de usuario con informes largos, porque la interfaz puede mostrar el estado del trabajo en lugar de quedarse congelada.

La automatización de Office debe sustituirse. Instalar Office en un servidor y automatizar Excel o Word desde IIS puede parecer fácil, pero no es seguro en un servidor. Usa bibliotecas diseñadas para generar y manipular documentos sin intervención, como EPPlus, Aspose, Syncfusion, DevExpress, Open XML SDK u otra biblioteca aprobada, según la licencia y los requisitos de los documentos.

```
private void exportButton_Click(object sender, EventArgs e)
{
    var rows = orderService.Search(currentFilter);
    var path = reportService.CreateOrdersWorkbook(rows);
    // In a real app, use the organization's approved download helper.
    Application.Download(path, "Orders.xlsx");
}
```



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Clasificar las operaciones con archivos como solo de servidor, mediadas por el cliente, de carga/descarga o no admitidas sin rediseño.
- ✓ Sustituir el uso del registro por base de datos, configuración de usuario o almacenamiento del navegador.
- ✓ Convertir los flujos de informes en generación en el servidor, con salida PDF mostrada en PDFViewer u ofrecida como descarga.
- ✓ Sustituir Office Interop/Automation por API de documentos administradas y seguras para el servidor.



Claves de la migración

- ✓ El código de archivos migrado se ejecuta en el servidor, salvo que se use carga/descarga o la mediación del sistema de archivos del cliente.
- ✓ En despliegues web o de servidor, la automatización de Office debe sustituirse por bibliotecas de documentos seguras para el servidor.
- ✓ Normalmente, los informes deben generar documentos en el servidor, como PDF, y mostrarlos o descargarlos mediante flujos seguros para el navegador.

Archivos, informes, automatización de Office y límites del navegador

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Sustituye un flujo de escritorio de exportación a Excel por un archivo generado en el servidor y descargado desde el navegador; añade un trabajo de informe en cola.

Pasos del laboratorio

- ✓ Localiza un flujo de importación/exportación de archivos y un flujo de informes.
- ✓ Sustituye las suposiciones sobre archivos locales por carga/descarga o almacenamiento en el servidor.
- ✓ Convierte la salida del informe en un PDF o en la descarga de un documento generado.
- ✓ Añade una cola de trabajos sencilla para los informes largos.
- ✓ Elimina la automatización directa de Office de la ruta migrada.
- ✓ Prueba en un entorno limpio similar a un servidor.

Modernización, interfaz adaptable, seguridad, despliegue y proyecto final

Convierte el port que ya funciona en una aplicación web Wisej.NET pulida, segura y lista para desplegar. Por qué importa, la lectura principal y los objetivos de aprendizaje: Léelo antes del laboratorio.



Enfoque de la migración

Convierte el port que ya funciona en una aplicación web Wisej.NET pulida, segura y lista para desplegar.

Por qué importa este módulo

Tras el primer port con éxito, los equipos pueden modernizar sin perder el control del proyecto. Los estudiantes aplican temas, propiedades adaptables, perfiles de cliente, formato HTML con cuidado, revisión de seguridad, empaquetado para el despliegue y una lista final de comprobación de preparación.

Lectura principal

Un primer port con éxito no es el final. Es el momento en que la modernización se vuelve más segura. Ahora el equipo puede mejorar áreas concretas sin desestabilizar toda la aplicación. Los temas, las propiedades adaptables, los perfiles de cliente, los controles mejorados y las notificaciones nativas de la web de Wisej.NET permiten al equipo modernizar de forma incremental.

El comportamiento adaptable es un tema de migración porque la mayoría de las pantallas WinForms se diseñaron para resoluciones fijas. Los perfiles de cliente de Wisej.NET permiten a la aplicación asociar valores de propiedad distintos a distintos dispositivos y tamaños de pantalla. Un control puede ser visible en el escritorio y estar oculto en el teléfono, un botón puede mostrar icono y texto en el escritorio y solo el icono en pantallas pequeñas, y una pantalla puede recolocar controles sin mantener bases de código separadas.

La revisión de seguridad debe cubrir tanto los riesgos antiguos como los nuevos. Entre los riesgos antiguos están la inyección SQL, el almacenamiento inseguro y los roles con demasiados permisos. Los nuevos riesgos propios de la web incluyen exponer operaciones del servidor al navegador, usar AllowHtml con valores introducidos por el usuario, las descargas de archivos confidenciales y los errores de sesión y autenticación. Wisej.NET codifica el texto de forma predeterminada, pero AllowHtml y el contenido de mensajes con HTML habilitado deben tratarse con cuidado.

El despliegue debe planificarse pronto. Debes saber si el destino es IIS en Windows, Linux, Docker o una plataforma en la nube. Confirma las versiones de los paquetes, las cadenas de conexión, las rutas de almacenamiento de archivos, las dependencias de fuentes de los informes, los permisos de la carpeta temporal, el registro, las comprobaciones de estado, HTTPS, el tiempo de espera de la sesión y los requisitos de balanceo de carga y afinidad de sesión.

Lista final de comprobación de preparación

- ✓ La aplicación arranca mediante los archivos de arranque de Wisej.NET y muestra la vista principal correcta.

- ✓ No queda estado por usuario en campos estáticos inseguros.
- ✓ Se han revisado todas las suposiciones sobre registro, archivos locales, automatización de Office y lanzamiento de procesos.
- ✓ Las cuadrículas grandes se han probado con un número de filas similar al de producción.
- ✓ Los diálogos transitorios se liberan.
- ✓ AllowHtml solo se usa con contenido de confianza o saneado.
- ✓ El comportamiento adaptable se ha probado en tamaños de escritorio, tableta y teléfono.
- ✓ La configuración de despliegue, los secretos, los registros, las rutas temporales y las comprobaciones de estado están documentados.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Aplicar temas y mixins sin reescribir las pantallas.
- ✓ Usar ClientProfiles.json y las propiedades adaptables para adaptar una interfaz de escritorio fija a tabletas y pantallas pequeñas.
- ✓ Aplicar reglas de seguridad para AllowHtml, la entrada del usuario, la autenticación y la validación de la sesión.
- ✓ Preparar el despliegue en IIS, Linux, Docker o la nube con notas sobre configuración, registro y balanceo de carga.



Claves de la migración

- ✓ La modernización debe venir después de una base de paridad que funcione.
- ✓ Los perfiles de cliente y las propiedades adaptables permiten que una pantalla migrada de resolución fija se adapte a las condiciones del dispositivo y del navegador.
- ✓ AllowHtml solo debe usarse con contenido de confianza o saneado, porque la codificación de texto predeterminada es más segura para la entrada del usuario.

Modernización, interfaz adaptable, seguridad, despliegue y proyecto final

Qué vas a construir en el laboratorio práctico, los pasos del laboratorio y cómo verificar el flujo migrado.



Laboratorio práctico

Termina el proyecto final: migra la aplicación, moderniza una pantalla, ejecuta las comprobaciones de seguridad y despliegue y presenta el informe de migración.

Pasos del laboratorio

- ✓ Aplica un tema o un mixin.
- ✓ Crea o actualiza ClientProfiles.json.
- ✓ Configura las propiedades adaptables de la pantalla migrada.
- ✓ Revisa el uso de AllowHtml y sanea el contenido o desactívalo donde haga falta.
- ✓ Prepara la configuración de despliegue y el perfil de publicación.
- ✓ Haz la demostración del proyecto final y entrega el informe de migración.