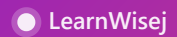


Temas, recursos, localización y modernización de la interfaz

Profundiza en los temas: cambio claro/oscuro en tiempo de ejecución, etiquetas de estado reutilizables mediante CSSStyle, gestión de recursos y localización sensible a la referencia cultural.



Objetivo aplicado

Modernizar de principio a fin el dashboard de operaciones de TicketOps: cambiar entre un tema claro y uno oscuro en tiempo de ejecución, construir un control reutilizable de chip de estado con estilo mediante CSSStyle, gestionar iconos e imágenes como recursos y localizar etiquetas y formatos para dos referencias culturales.

1. Cambiar de tema en tiempo de ejecución con Application.Theme

TicketOps Console incluye un interruptor en la cabecera para que los operadores puedan pasar de un tema claro para los turnos de día a un tema oscuro para el puesto de operaciones nocturno. Asigna un nombre de tema a `Application.Theme` y Wisej.NET vuelve a renderizar todas las ventanas abiertas; no reconstruyes los controles tú mismo.

Mantén los nombres de los temas en un solo lugar y guarda la elección del operador para que el dashboard vuelva a abrirse con el mismo tema tras refrescar la sesión.

```
// TicketOps header: a single toggle drives the whole console
private void themeSwitch_CheckedChanged(object sender, EventArgs e)
{
    // Built-in Wisej.NET theme names shipped with the app
    var theme = themeSwitch.Checked ? "BootstrapDark-4" : "Bootstrap-4";

    Application.Theme = theme; // re-themes all open windows live
    Application.Session["ticketops.theme"] = theme; // remember per session
}

// On startup, restore the last theme before the dashboard is shown
protected override void OnLoad(EventArgs e)
{
    if (Application.Session["ticketops.theme"] is string saved)
        Application.Theme = saved;

    base.OnLoad(e);
}
```



Regla de producción

Nunca repartas nombres de temas como literales por las pantallas. Mantén `"Bootstrap-4"` y `"BootstrapDark-4"` en una constante para que un cambio de marca sea una modificación de una línea.

2. Un chip de estado reutilizable con estilo mediante `CSSStyle`, no con colores fijos

Cada fila de Work Orders muestra un estado: *Open*, *In Progress*, *Blocked*, *Resolved*. En lugar de colorear etiquetas a mano en cada pantalla, construye un único UserControl `StatusChip`. Expone una propiedad `Status` y aplica un token de clase `CSSStyle`; los colores reales viven en el tema, así que tanto el tema claro como el oscuro renderizan bien el chip.

```
// StatusChip: a reusable UserControl for the operations dashboard
public partial class StatusChip : UserControl
{
    private WorkOrderStatus _status;

    public WorkOrderStatus Status
    {
        get => _status;
        set
        {
            _status = value;
            labelText.Text = value.ToString();

            // Drive appearance through a CSSStyle token, not inline colors
            // Theme defines .chip-open, .chip-blocked, etc. for each theme
            this.CssClass = "chip chip-" + value.ToString().ToLowerInvariant();
        }
    }
}
```

Como el chip hace referencia a un nombre de clase y el tema aporta la paleta, el mismo control se ve bien de día y de noche sin un solo condicional en tu código.

3. Elegir dónde va cada estilo

El error de modernización más común es poner el color en la capa equivocada. Usa esta tabla para decidir dónde vive cada regla visual antes de escribirla.

Técnica	Alcance	Úsala para	Evítala para
Tema / Theme Builder	Toda la aplicación	Paleta, tipografía, aspecto por defecto de los controles, colores de los chips	Retoques puntuales en un único control
CSSStyle / CssClass	Un control o un UserControl reutilizable	Variantes de componentes compartidos, como el chip de estado	Decisiones de color de toda la aplicación
BackColor / ForeColor en línea	Una instancia de control	Casi nada en las pantallas modernas	Cualquier cosa a la que tengas que cambiar el tema más adelante



Trampa: colores fijos en el código

Un chip configurado con `chip.BackColor = Color.Red` sigue rojo con el tema oscuro y rompe el contraste. Canaliza el aspecto a través del tema o de una `CssClass` para que ambos temas sigan siendo correctos.

4. Gestionar iconos e imágenes como recursos

La barra de herramientas del dashboard necesita un icono de actualizar, un icono de exportar y un glifo de estado por chip. Trátalos como recursos gestionados y no como rutas de archivo sueltas. Wisej.NET puede resolver imágenes que dependen del tema, y reutilizar una única referencia de recurso mantiene coherente la barra de herramientas en todas las pantallas.

Tipo de recurso	Ideal para	Cómo se hace referencia a él
Mixin de imágenes del tema	Iconos que cambian según el tema (glifos claros frente a oscuros)	Por el nombre de imagen del tema, resuelto automáticamente
Recurso incrustado / de proyecto	Logotipos e imágenes que nunca cambian según el tema	A través del accesor de recursos generado
Archivo externo / URL	Contenido aportado por el usuario, como adjuntos	Por ruta o URL en tiempo de ejecución

```
// Toolbar buttons pull from managed resources, not scattered paths
refreshButton.ImageSource = "resource.wx/TicketOps.Icons.refresh.svg";
exportButton.ImageSource = "resource.wx/TicketOps.Icons.export.svg";

// One reference reused everywhere keeps the toolbar consistent
public static class Glyphs
{
    public const string Refresh = "resource.wx/TicketOps.Icons.refresh.svg";
    public const string Export = "resource.wx/TicketOps.Icons.export.svg";
}
```

5. Localizar etiquetas y formatos para dos referencias culturales

TicketOps da servicio a equipos de operaciones en inglés y en italiano. Hay dos cosas que deben cambiar a la vez: el **texto** de las etiquetas (mediante recursos localizables) y el **formato** de fechas y moneda (mediante `CultureInfo`). Localizar una sin la otra produce palabras en inglés junto a números con formato italiano, y eso parece roto.

Marca como localizable el texto de formularios y controles para que los archivos `.resx` traducidos aporten cada referencia cultural. Para los valores, formatea explícitamente con la referencia cultural del operador en lugar de depender de la predeterminada del servidor.

```
// Culture-aware formatting for the Work Order detail panel
using System.Globalization;

// en-US → 7/3/2026, $1,250.00   it-IT → 03/07/2026, 1.250,00 €
var culture = new CultureInfo(operatorCultureCode); // "en-US" or "it-IT"

createdLabel.Text = workOrder.CreatedOn.ToString("d", culture);
costLabel.Text    = workOrder.LaborCost.ToString("C", culture);
slaLabel.Text     = workOrder.DueBy.ToString("g", culture);

// Set the request culture so localizable resources resolve too
Application.CurrentCulture = culture;
Application.CurrentUICulture = culture;
```



Regla de producción

Establece a la vez `CurrentUICulture` para el texto y `CurrentCulture` para el formato, y vuelve a probar la etiqueta más ancha. Los textos en italiano suelen ser más largos que en inglés y pueden recortarse en un chip o un botón de ancho fijo.

6. Una lista de comprobación de modernización de la interfaz aplicada al dashboard

Repasa cada pantalla de operaciones con estas preguntas antes de considerarla moderna. Convierten las reglas abstractas de la Guía de la lección en comprobaciones concretas sobre pantallas reales de TicketOps.

- ✓ ¿Cada color de estado viene del tema o de una `CssClass`, sin ningún `BackColor` en línea?
- ✓ ¿El interruptor claro/oscuro cambia el tema de toda la consola, incluidos los chips y los iconos de la barra de herramientas?
- ✓ ¿Todos los iconos de la barra de herramientas se resuelven desde recursos gestionados en lugar de rutas sueltas?
- ✓ ¿Cada etiqueta visible es localizable, y el panel formatea fechas y moneda según `CultureInfo`?
- ✓ ¿La etiqueta traducida más ancha sigue cabiendo sin recortarse en ambos temas?
- ✓ ¿Podría un futuro cambio de marca modificar la paleta sin editar pantallas individuales?

Vocabulario clave

- ✓ **Application.Theme:** asigna un nombre de tema para volver a renderizar todas las ventanas abiertas en tiempo de ejecución.
- ✓ **CssClass / CSSStyle:** asocia un token de clase para que un control reutilizable herede los colores del tema.
- ✓ **Chip de estado:** un único UserControl que expresa el estado de una Work Order mediante una clase, no con color en línea.
- ✓ **Recurso gestionado:** un icono o imagen referenciado por nombre para que se mantenga coherente y adaptado al tema.
- ✓ **Propiedad localizable:** texto de un control que procede de recursos `.resx` específicos de cada referencia cultural.
- ✓ **CultureInfo:** determina el formato de fechas y moneda adaptado a la referencia cultural de cada equipo de operaciones.