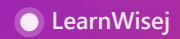


Proyecto final listo para móvil: accesibilidad, rendimiento y gobernanza en producción

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Terminar Adaptive Operations Console como un conjunto de pantallas de escritorio, tableta y teléfono listo para producción, revisar el foco, el contraste, los estados no válidos y los objetivos táctiles, mantener los manejadores de perfil ligeros e idempotentes, y gobernar el tema, el CSS y ClientProfiles.json como código. Qué significa esto para un desarrollador de Wisej.NET y cómo leer este módulo.



Enfoque de producción

Terminar Adaptive Operations Console como un conjunto de pantallas de escritorio, tableta y teléfono listo para producción, revisar el foco, el contraste, los estados no válidos y los objetivos táctiles, mantener los manejadores de perfil ligeros e idempotentes, y gobernar el tema, el CSS y ClientProfiles.json como código.

1. El problema en producción

Seis módulos de trabajo conviven ahora en una sola solución: un tema derivado de un tema base, CSS con ámbito, un shell de contenedores anidados, regiones Flow, Table y Flex, un ClientProfiles.json y un manejador de perfil. Lo que convierte eso en un producto no es otra funcionalidad. Es la prueba de que la consola se puede usar en todos los perfiles, de que un usuario de teclado o táctil puede manejarla, de que un redimensionado no desencadena reconstrucciones costosas, y de que el siguiente desarrollador, o la siguiente actualización de Wisej.NET, no puede romper sin avisar el sistema visual. Este módulo es esa prueba, junto con el paquete de contexto que permite a un asistente de IA responder preguntas sobre las decisiones que tomaste.

2. Conceptos clave

Concepto	Por qué importa
conjunto de pantallas del proyecto final	Adaptive Operations Console terminada: tema personalizado, appearances semánticas, CSS con ámbito, un shell de escritorio, un shell adaptable para tableta, un shell para teléfono, una cuadrícula, un editor, mensajes de validación, navegación y un indicador de perfil visible en modo de prueba.
revisión de accesibilidad visual	Una revisión de los marcos de foco, los estados deshabilitados y no válidos, el contraste, el espaciado táctil, los tooltips de los botones solo icono y el orden de tabulación con teclado, hecha manejando la pantalla y no leyendo el tema.
manejador de perfil idempotente	Un manejador <code>ResponsiveProfileChanged</code> que asigna valores finales para el perfil actual, de modo que ejecutarlo dos veces, o después de una recarga, produce exactamente la misma interfaz.
estabilidad del diseño	<code>MinimumSize</code> y <code>MaximumSize</code> evitan que las regiones se colapsen, y <code>AutoSize</code> se mantiene fuera de las jerarquías grandes de contenedores, de modo que los redimensionados son predecibles y ligeros.
gobernanza del tema	El JSON del tema, la hoja de estilos con ámbito y <code>ClientProfiles.json</code> están bajo control de código fuente, se revisan como diffs, se documentan como tokens y nunca se editan dentro de un tema integrado.
personalización segura ante actualizaciones	Las appearances documentadas, los flujos del Theme Builder y las propiedades del servidor sobreviven a las actualizaciones; los selectores CSS que dependen del DOM interno de un widget pueden no hacerlo, así que se limitan en ámbito, se documentan o se evitan.
matriz de QA responsive y paquete de contexto	La matriz registra anchos, orientaciones, perfil esperado y comprobaciones por escenario; el paquete de contexto registra el árbol de decisión, las reglas de respuesta y el mapa de fuentes a partir de los que responde un asistente.

3. Desarrollo de la lección

Una consola, tres shells, un sistema visual

El proyecto final se evalúa por su coherencia, no por su novedad. Los controles estándar toman su identidad del tema `AdaptiveOps`, que deriva de un tema base y define tokens de color con nombre para la marca, las superficies, el texto, `danger`, `warning`, `success` y el marco de foco. Las variantes de negocio son appearances semánticas asignadas mediante `AppearanceKey` : `action-button` , `destructive-button` , `kpi-card` , `nav-item` , `compact-editor` o equivalentes, al menos cuatro. El contenido específico de la aplicación, como las tarjetas de métricas, usa clases con ámbito mediante `CssClass` , y `CssStyle` aparece solo donde un valor se calcula en tiempo de ejecución. Si una pantalla recurre directamente a `BackColor` , `ForeColor` o `Font` , la revisión pregunta cuál de esas capas debería haber sido la responsable.

El shell mantiene la estructura que construiste en los módulos 4 y 5: barra de herramientas acoplada, carril de navegación, área de trabajo, región de detalles y pie de estado, con un `FlowLayoutPanel` para los

filtros y las tarjetas, un `TableLayoutPanel` para el editor de tickets alineado y un `FlexLayoutPanel` para el área de trabajo proporcional. Las diferencias entre escritorio, tableta y teléfono vienen de `ClientProfiles.json` junto con propiedades responsive asignadas en el diseñador; el único manejador `ResponsiveProfileChanged` existe para el comportamiento que una propiedad no puede expresar, como abrir el editor de detalles como formulario modal en el teléfono. El indicador de perfil se queda en la región de estado, visible en modo de prueba, para que cada captura de la matriz de QA nombre el perfil que la produjo.

Accesibilidad que puedes verificar usando la pantalla

Un tema debe conservar un foco visible, estados deshabilitados claros, texto legible, contraste suficiente, indicación de estados no válidos y un espaciado apto para uso táctil. La regresión más habitual es eliminar el marco de foco porque parece anticuado; la regla es sustituirlo por uno mejor basado en el token `focusFrame`, nunca eliminarlo. Los estados no válidos y de advertencia no deben depender solo del color: acompaña el borde danger con un icono o un mensaje que el usuario pueda leer, para que el estado se perciba también con daltonismo o en un proyector. El contraste se comprueba con los pares de tokens que usa realmente el tema, por ejemplo `textMuted` sobre `surfaceAlt`, no con el color de marca de forma aislada.

Los botones solo icono de la barra de herramientas, a los que cambia el perfil de tableta mediante el valor responsive `Display`, necesitan `ToolTipText` o una etiqueta accesible para seguir siendo identificables. El orden de tabulación se verifica recorriendo la pantalla con el teclado y confirmando que `TabStop` y `TabIndex` sitúan el comando principal donde el usuario lo espera. Por último, el contenido oculto necesita una explicación: cuando el carril de navegación y el panel de detalles se ocultan en el teléfono, las funciones que ofrecían deben seguir siendo accesibles, a través del botón de menú móvil y del editor modal, y la nota de arquitectura registra esa alternativa para cada control oculto.

Manejadores ligeros, diseños estables, archivos gobernados

El diseño responsive se vuelve costoso cuando cada redimensionado provoca un diseño complejo, recargas de datos y actualizaciones del cliente. El manejador que reacciona a un cambio de perfil debe ser idempotente y ligero: asigna la visibilidad, el acoplamiento y el tamaño finales para el perfil actual en lugar de alternarlos, no reconstruye la interfaz cuando bastan unos pocos cambios de propiedades, y nunca recarga datos solo porque haya cambiado el ancho. El método `ApplyProfile` del laboratorio se llama una vez desde el constructor con `Application.ActiveProfile` y otra desde el manejador con el perfil actual del evento, y la suscripción se elimina cuando se libera la página:

```
private void ApplyProfile(string name)
{
    if (name == _lastProfile) return;           // same profile: no work
    _lastProfile = name;
    bool phone = name != null && name.StartsWith("Phone");

    lblProfile.Text = name ?? "Default";        // test-mode indicator
    navRail.Visible = !phone;                   // final values, never toggles
    detailsPanel.Visible = !phone;
    btnOpenDetails.Visible = phone;
    ticketGrid.MinimumSize = new Size(phone ? 0 : 320, 240);
}
```

La estabilidad del diseño nace de la misma disciplina: `MinimumSize` y `MaximumSize` en las regiones acopladas y con `FillWeight` para que nada se reduzca a cero, nada de `AutoSize` en jerarquías grandes de contenedores, y cambios de datos responsive intencionados en lugar de efectos secundarios de un redimensionado. Después, trata el tema como código. `AdaptiveOps.theme`, `AdaptiveOps.css` y `ClientProfiles.json` están en el control de código fuente con nombres claros; sus diffs se revisan; los tokens de diseño están documentados; los temas integrados nunca se editan directamente. El equipo deja por escrito cuándo puede un desarrollador usar `BackColor`, `ForeColor` o `Font`, cuándo `AppearanceKey`, `CssClass`, `CssStyle` o CSS personalizado son la herramienta adecuada, y mantiene un conjunto de capturas de las pantallas principales en tamaños de escritorio, tableta y teléfono.

La seguridad ante actualizaciones se deriva de las mismas reglas. Las appearances documentadas del tema, los flujos del Theme Builder y las propiedades del servidor son estables entre versiones de Wisej.NET; un selector CSS que depende del DOM interno de un widget no lo es. Cuando un widget personalizado o de terceros necesite de verdad CSS directo, limita estrictamente el ámbito del selector y documenta por qué el sistema de temas no es responsable de esa pieza. Después de cada actualización, vuelve a ejecutar la matriz de capturas responsive e inspecciona cada clave de appearance personalizada y cada estado específico de widget.



Errores habituales

Eliminar los marcos de foco en lugar de rediseñarlos. Señalar el estado no válido solo con color. Un manejador de perfil que alterna estados o reconstruye toda la página en cada cambio. Regiones con `FillWeight` sin `MinimumSize`. Editar directamente un tema integrado. CSS global aplicado a controles estándar mediante selectores del DOM interno. Un conjunto de capturas que se hace una vez y nunca se repite tras una actualización.

Un paquete de contexto a partir del que pueda responder un asistente

El último entregable está escrito para dos lectores: el siguiente desarrollador y un asistente de IA con contexto. La nota de arquitectura explica cada decisión de tema, diseño y perfil y qué mecanismo es responsable de ella. El paquete de contexto añade las reglas de respuesta que debe seguir el asistente: responder primero a partir del curso y en segundo lugar a partir de la documentación oficial del mapa de fuentes; advertir contra los selectores internos frágiles y contra cambiar un tema compartido para todas las sesiones cuando se pretendía cambiarlo para una; no proponer nunca media queries CSS como

mecanismo responsive principal cuando encajan las propiedades responsive o ClientProfile; y preguntar por el perfil y el tamaño de pantalla cuando un problema responsive sea ambiguo. El mapa de fuentes enumera las páginas de documentación que respaldan cada regla para que el asistente pueda citarlas en lugar de adivinar.

En el centro del paquete está el árbol de decisión del curso, aplicado en orden hasta que un escalón encaja:

- Primero el tema, para la identidad de los controles estándar.
- `AppearanceKey` para las variantes semánticas.
- `CssClass` para el CSS de la aplicación con ámbito y los widgets de terceros.
- `CssStyle` solo para valores dinámicos puntuales.
- Contenedores de diseño para el posicionamiento adaptable.
- Propiedades responsive para los valores por perfil definidos en el diseñador.
- `ResponsiveProfileChanged` para el comportamiento en tiempo de ejecución.
- ClientProfiles.json para las propias definiciones de los perfiles.
- Gobernanza de producción para una entrega mantenible.

La rúbrica pondera el diseño y la coherencia del tema con un 25 por ciento, la corrección del diseño responsive con un 25, ClientProfile y las propiedades responsive con un 20, la mantenibilidad y la gobernanza con un 15, y la explicación y la alineación del contexto con un 15. Esos últimos treinta puntos se ganan con la nota y el paquete: si el asistente, leyendo solo lo que escribiste, puede explicar por qué la región de detalles es un modal en el teléfono y por qué la tarjeta de métricas es una `CssClass` y no una `appearance`, la consola está terminada.



Regla del sistema visual

Usa el mecanismo de Wisej.NET responsable del problema: el tema para la identidad de los controles estándar, `AppearanceKey` para las variantes semánticas, `CssClass` para CSS específico de la aplicación con ámbito, `CssStyle` solo para valores dinámicos puntuales, contenedores de diseño en lugar de código de redimensionado, propiedades responsive para los valores por perfil definidos en el diseñador, y `ResponsiveProfileChanged` para el comportamiento que no es simplemente visual.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Entregar una interfaz de marca para escritorio, tableta y teléfono que use el tema, los contenedores de diseño y ClientProfile de forma coherente.
- ✓ Revisar la accesibilidad visual: indicadores de foco, estados no válidos, contraste, objetivos táctiles, orden de tabulación y comportamiento del contenido oculto.
- ✓ Preparar una lista de verificación de producción para archivos de tema, archivos CSS, ClientProfiles.json, capturas, pruebas de aceptación y personalizaciones seguras ante actualizaciones.

- ✓ Empaquetar el mapa de fuentes y los materiales RAG para que un asistente de IA con contexto pueda responder preguntas de implementación a partir del curso.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Completa el proyecto final de Adaptive Operations Console: un tema personalizado con tokens con nombre y al menos cuatro appearances semánticas, uso de `CssClass` con ámbito, un shell de escritorio con regiones `Flow`, `Table` y `Flex`, un `ClientProfiles.json` con comportamiento para teléfono, tableta, escritorio pequeño y escritorio, propiedades responsive junto con un manejador `ResponsiveProfileChanged` idempotente, y un indicador de perfil visible en modo de prueba. Realiza la revisión de accesibilidad de marcos de foco, estados no válidos, contraste, objetivos táctiles y orden de tabulación, completa la matriz de QA responsive con capturas para cada perfil, y redacta la nota de arquitectura y el paquete de contexto que expliquen cada decisión de tema, diseño y perfil.

Punto de partida

Abre la solución Adaptive Operations Console que has ido construyendo en los módulos 1 a 6: existen `AdaptiveOps.theme` y `AdaptiveOps.css` con ámbito, el shell está formado por contenedores anidados con regiones `Flow`, `Table` y `Flex`, `ClientProfiles.json` enumera seis perfiles y el manejador del Módulo 6 ya muestra el nombre del perfil activo en la etiqueta de estado. En este laboratorio no se construye nada nuevo desde cero; cada paso refuerza, revisa o documenta lo que ya existe.

Enfoque sugerido

- ✓ Revisa primero las capas visuales: confirma que el tema deriva de un tema base con tokens con nombre, cuenta las appearances semánticas asignadas mediante `AppearanceKey` (añade una `destructive-button`, `kpi-card` o `nav-item` si tienes menos de cuatro) y traslada cualquier sobrescritura suelta de `BackColor`, `ForeColor` o `Font` al tema o a una `CssClass`.
- ✓ Refuerza el diseño: establece `MinimumSize` en cada región acoplada o con `FillWeight` y `MaximumSize` donde una región no deba crecer, quita `AutoSize` de las jerarquías grandes de contenedores y redimensiona pasando por todos los anchos para confirmar que nada se colapsa ni se desborda horizontalmente.
- ✓ Reescribe la lógica de perfiles como un único método `ApplyProfile(string name)` que asigne valores finales y salga de inmediato cuando el perfil no haya cambiado; llámalo desde el constructor con `Application.ActiveProfile` y desde el manejador `ResponsiveProfileChanged`, cancela la suscripción en `Dispose` y muestra `lblProfile` solo cuando esté activado un indicador de modo de prueba.
- ✓ Realiza la revisión de accesibilidad usando la pantalla: recorre con `Tab` todas las regiones y corrige `TabIndex` y `TabStop`, lanza la validación y confirma que el estado no válido tiene un icono o un mensaje además del color, comprueba el contraste de cada par de tokens, añade `ToolTipText` a los botones solo icono y mide los objetivos táctiles con ancho de teléfono.
- ✓ Completa la matriz de QA responsive (escritorio, escritorio pequeño, tableta en vertical y en horizontal, teléfono en vertical y en horizontal) con el perfil esperado, una captura y las comprobaciones de cada fila; después redacta la nota de arquitectura y el paquete de contexto:

árbol de decisión, reglas de respuesta, mapa de fuentes y una pregunta que el asistente deba responder a partir de tu material.

Entregables requeridos

- ✓ Adaptive Operations Console terminada con tema personalizado, appearances semánticas, CSS con ámbito y regiones Flow, Table y Flex
- ✓ ClientProfiles.json, propiedades responsive y un manejador ResponsiveProfileChanged idempotente con un indicador de perfil en modo de prueba
- ✓ Revisión de accesibilidad que cubre marcos de foco, estados no válidos, contraste, objetivos táctiles y orden de tabulación
- ✓ Matriz de QA responsive con capturas de escritorio, tableta y teléfono en ambas orientaciones
- ✓ Nota de arquitectura y paquete de contexto para IA que explican cada decisión de tema, diseño y perfil

Preguntas de repaso

- ✓ Tu manejador `ResponsiveProfileChanged` se ejecuta dos veces para el perfil Tablet después de recargar el navegador. ¿Qué línea tuya garantiza que la segunda ejecución no cambie nada, y qué vería el usuario si faltara esa línea?
- ✓ Un revisor te pide quitar el marco de foco de `action-button` porque «parece anticuado». ¿Qué cambias en su lugar, qué token lo controla y qué otros dos estados vuelves a verificar después en el Theme Builder?
- ✓ El encabezado de la cuadrícula de tickets necesita un ajuste al que solo llega un selector sobre el DOM interno del widget. ¿Qué pruebas primero, qué debe cumplirse antes de permitir el selector, y qué dejas por escrito para que la siguiente actualización no sorprenda al equipo?

Entrega del proyecto final

Este laboratorio es también la entrega del proyecto final del curso. Entrega el proyecto fuente, AdaptiveOps.theme, AdaptiveOps.css, ClientProfiles.json, la matriz de QA responsive completa con sus capturas, la revisión de accesibilidad, la nota de arquitectura y el paquete de contexto. Prepárate para recorrer la consola en cada perfil y para responder, sobre cualquier control en pantalla, qué mecanismo es responsable de su aspecto y por qué.

Área de la rúbrica	Peso	Así es un trabajo excelente
Diseño y coherencia del tema	25%	Tema derivado de un tema base, tokens reutilizados en todas partes, appearances semánticas para cada variante de negocio, sin sobrescrituras improvisadas de color o fuente.
Corrección del diseño responsive	25%	Los contenedores se encargan del diseño, las regiones tienen tamaños mínimos, cada ancho de la matriz de QA se puede usar en ambas orientaciones.
ClientProfile y propiedades responsive	20%	Perfiles en orden de primera coincidencia, primero los valores asignados en el diseñador, un único manejador idempotente para cambios de comportamiento reales, indicador en modo de prueba.
Mantenibilidad y gobernanza	15%	Archivos en el control de código fuente, diffs revisados, excepciones de CSS documentadas, matriz de capturas y revisión de accesibilidad completas y repetibles tras las actualizaciones.
Explicación y alineación del contexto para IA	15%	Nota de arquitectura y paquete de contexto a partir de los cuales un asistente puede explicar cada decisión y advertir sobre las elecciones frágiles.



Regla del sistema visual

Usa el mecanismo de Wisej.NET responsable del problema: el tema para la identidad de los controles estándar, AppearanceKey para las variantes semánticas, CssClass para CSS específico de la aplicación con ámbito, CssStyle solo para valores dinámicos puntuales, contenedores de diseño en lugar de código de redimensionado, propiedades responsive para los valores por perfil definidos en el diseñador, y ResponsiveProfileChanged para el comportamiento que no es simplemente visual.