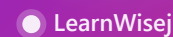


Capstone pronto per il mobile, accessibilità, prestazioni e governance di produzione

Guida della lezione · Guida al lab / esame



Guida della lezione

Completare Adaptive Operations Console come set di schermate desktop, tablet e telefono pronto per la produzione, rivedere focus, contrasto, stati non validi e aree di tocco, mantenere i gestori di profilo leggeri e idempotenti, e governare il tema, il CSS e ClientProfiles.json come codice. Che cosa significa per uno sviluppatore Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Completare Adaptive Operations Console come set di schermate desktop, tablet e telefono pronto per la produzione, rivedere focus, contrasto, stati non validi e aree di tocco, mantenere i gestori di profilo leggeri e idempotenti, e governare il tema, il CSS e ClientProfiles.json come codice.

1. Il problema in produzione

Sei moduli di lavoro vivono ora in un'unica soluzione: un tema derivato da un tema di base, CSS circoscritto, uno shell di contenitori annidati, regioni Flow, Table e Flex, un ClientProfiles.json e un gestore del profilo. Ciò che trasforma tutto questo in un prodotto non è un'altra funzionalità. È la prova che la console è utilizzabile su ogni profilo, che un utente da tastiera o touch può usarla, che un ridimensionamento non scatena ricostruzioni costose, e che il prossimo sviluppatore, o il prossimo aggiornamento di Wisej.NET, non può rompere in silenzio il sistema visivo. Questo modulo è quella prova, più il grounding pack che permette a un assistente IA di rispondere a domande sulle decisioni che hai preso.

2. Concetti chiave

Concetto	Perché è importante
set di schermate del capstone	Adaptive Operations Console completa: tema personalizzato, appearance semantiche, CSS circoscritto, uno shell desktop, uno shell tablet adattivo, uno shell per telefono, una griglia, un editor, feedback di validazione, navigazione e un indicatore del profilo visibile in modalità di test.
revisione dell'accessibilità visiva	Un passaggio su focus frame, stati disabilitati e non validi, contrasto, spaziatura per il tocco, tooltip sui pulsanti solo icona e ordine di tabulazione da tastiera, fatto usando davvero la schermata invece di leggere il tema.
gestore di profilo idempotente	Un gestore <code>ResponsiveProfileChanged</code> che assegna valori finali per il profilo corrente, così eseguirlo due volte, o dopo un ricaricamento, produce esattamente la stessa UI.
stabilità del layout	<code>MinimumSize</code> e <code>MaximumSize</code> impediscono alle regioni di collassare, e <code>AutoSize</code> resta fuori dalle grandi gerarchie di contenitori, così i ridimensionamenti sono prevedibili e leggeri.
governance del tema	Il JSON del tema, il foglio di stile circoscritto e ClientProfiles.json sono nel controllo del codice sorgente, revisionati come diff, documentati come token e mai modificati dentro un tema integrato.
personalizzazione sicura rispetto agli aggiornamenti	Appearance documentate, flussi del Theme Builder e proprietà lato server sopravvivono agli aggiornamenti; i selettori CSS che dipendono dal DOM interno di un widget potrebbero non farlo, quindi vanno circoscritti, documentati o evitati.
matrice di QA responsive e grounding pack	La matrice registra larghezze, orientamenti, profilo atteso e controlli per scenario; il grounding pack registra l'albero decisionale, le regole di risposta e la mappa delle fonti da cui risponde un assistente.

3. Il filo della lezione

Una console, tre shell, un sistema visivo

Il capstone viene giudicato sulla coerenza, non sulla novità. I controlli standard prendono la loro identità dal tema AdaptiveOps, che deriva da un tema di base e definisce token di colore con nome per brand, superfici, testo, danger, warning, success e focus frame. Le varianti di business sono appearance semantiche assegnate tramite `AppearanceKey`: `action-button`, `destructive-button`, `kpi-card`, `nav-item`, `compact-editor` o equivalenti, almeno quattro. I contenuti specifici dell'applicazione come le card delle metriche usano classi circoscritte tramite `CssClass`, e `CssStyle` compare solo dove un valore viene calcolato a runtime. Se una schermata ricorre direttamente a `BackColor`, `ForeColor` o `Font`, la revisione chiede quale di quei livelli avrebbe dovuto possedere la decisione.

Lo shell mantiene la struttura costruita nei Moduli 4 e 5: toolbar in dock, barra di navigazione, area di lavoro, regione dei dettagli e piè di pagina di stato, con una `FlowLayoutPanel` per filtri e card, una `TableLayoutPanel` per l'editor dei ticket allineato e una `FlexLayoutPanel` per l'area di lavoro

proporzionale. Le differenze tra desktop, tablet e telefono vengono da ClientProfiles.json più le proprietà responsive assegnate nel designer; l'unico gestore `ResponsiveProfileChanged` esiste per il comportamento che una proprietà non può esprimere, come aprire l'editor dei dettagli come form modale sul telefono. L'indicatore del profilo resta nella regione di stato, visibile in modalità di test, così ogni screenshot della matrice di QA indica il profilo che l'ha prodotto.

Un'accessibilità che puoi verificare usando la schermata

Un tema deve preservare il focus visibile, stati disabilitati chiari, testo leggibile, contrasto sufficiente, feedback sugli stati non validi e spaziature adatte al tocco. La regressione più comune è eliminare il focus frame perché sembra vecchio; la regola è sostituirlo con uno migliore disegnato a partire dal token `focusFrame`, mai rimuoverlo. Gli stati non validi e di avviso non devono affidarsi solo al colore: abbina il bordo danger a un'icona o a un messaggio che l'utente può leggere, così lo stato resiste al daltonismo e a un proiettore. Il contrasto si verifica sulle coppie di token che il tema usa davvero, per esempio `textMuted` su `surfaceAlt`, non sul colore del brand preso da solo.

I pulsanti della toolbar solo icona, a cui il profilo tablet passa tramite il valore responsive di `Display`, hanno bisogno di `ToolTipText` o di un'etichetta accessibile per restare individuabili. L'ordine di tabulazione si verifica percorrendo la schermata con la tastiera e confermando che `TabStop` e `TabIndex` mettano il comando principale dove l'utente se lo aspetta. Infine, i contenuti nascosti hanno bisogno di una storia: quando la barra di navigazione e il pannello dei dettagli sono nascosti sul telefono, le funzioni che contenevano devono restare raggiungibili, tramite il pulsante del menu mobile e l'editor modale, e la nota di architettura registra quell'alternativa per ogni controllo nascosto.

Gestori leggeri, layout stabili, file governati

Il design responsive diventa costoso quando ogni ridimensionamento provoca layout complessi, ricaricamenti di dati e aggiornamenti del client. Il gestore che reagisce a un cambio di profilo deve essere idempotente e leggero: assegna la visibilità, il docking e la dimensione finali per il profilo corrente invece di alternarli, non ricostruisce la UI quando bastano poche modifiche alle proprietà, e non ricarica mai i dati solo perché è cambiata la larghezza. Il metodo `ApplyProfile` del lab viene chiamato una volta dal costruttore con `Application.ActiveProfile` e di nuovo dal gestore con il profilo corrente dell'evento, e la sottoscrizione viene rimossa quando la pagina viene eliminata:

```
private void ApplyProfile(string name)
{
    if (name == _lastProfile) return;           // stesso profilo: nessun lavoro
    _lastProfile = name;
    bool phone = name != null && name.StartsWith("Phone");

    lblProfile.Text = name ?? "Default";         // indicatore della modalità di test
    navRail.Visible = !phone;                   // valori finali, mai alternanze
    detailsPanel.Visible = !phone;
    btnOpenDetails.Visible = phone;
    ticketGrid.MinimumSize = new Size(phone ? 0 : 320, 240);
}
```

La stabilità del layout viene dalla stessa disciplina: `MinimumSize` e `MaximumSize` sulle regioni con `FillWeight` e in dock così che nulla collassi a zero, niente `AutoSize` sulle grandi gerarchie di contenitori, e modifiche responsive ai dati intenzionali invece che effetti collaterali di un ridimensionamento. Poi tratta il tema come codice. `AdaptiveOps.theme`, `AdaptiveOps.css` e `ClientProfiles.json` vivono nel controllo del codice sorgente con nomi chiari; i loro diff vengono revisionati; i token di design sono documentati; i temi integrati non vengono mai modificati direttamente. Il team mette per iscritto quando uno sviluppatore può usare `BackColor`, `ForeColor` o `Font`, quando lo strumento giusto è `AppearanceKey`, `CssClass`, `CssStyle` o CSS personalizzato, e mantiene un set di screenshot delle schermate principali alle dimensioni desktop, tablet e telefono.

La sicurezza rispetto agli aggiornamenti discende dalle stesse regole. Le appearance documentate del tema, i flussi del Theme Builder e le proprietà lato server sono stabili tra le versioni di Wisej.NET; un selettore CSS che dipende dal DOM interno di un widget non lo è. Quando un widget di terze parti o personalizzato ha davvero bisogno di CSS diretto, circoscrivi strettamente il selettore e documenta perché il sistema dei temi non possiede quel pezzo. Dopo ogni aggiornamento, esegui di nuovo la matrice di screenshot responsive e ispeziona ogni chiave di appearance personalizzata e ogni stato specifico dei widget.



Errori tipici

Rimuovere i focus frame invece di ridisegnarli. Segnalare lo stato non valido solo con il colore. Un gestore di profilo che alterna lo stato o ricostruisce l'intera pagina a ogni cambio. Regioni con `FillWeight` senza un `MinimumSize`. Modificare sul posto un tema integrato. CSS globale applicato come pezza sui controlli standard tramite selettori sul DOM interno. Un set di screenshot che esiste una volta sola e non viene mai rieseguito dopo un aggiornamento.

Un grounding pack da cui un assistente può rispondere

L'ultimo deliverable è scritto per due lettori: il prossimo sviluppatore e un assistente IA con grounding. La nota di architettura spiega ogni decisione su tema, layout e profili e quale meccanismo la possiede. Il grounding pack aggiunge le regole di risposta che l'assistente deve seguire: rispondere prima a partire dal corso e poi dalla documentazione ufficiale elencata nella mappa delle fonti; mettere in guardia dai selettori interni fragili e dal cambiare un tema condiviso per ogni sessione quando si intendeva una sola sessione; non proporre mai le media query CSS come meccanismo responsive principale quando sono adatti le proprietà responsive o `ClientProfile`; e chiedere profilo e dimensione dello schermo quando un problema responsive è ambiguo. La mappa delle fonti elenca le pagine di documentazione dietro ogni regola così che l'assistente possa citare invece di indovinare.

Al centro del pack c'è l'albero decisionale del corso, applicato in ordine finché un gradino non si adatta:

- Prima il tema, per l'identità dei controlli standard.
- `AppearanceKey` per le varianti semantiche.
- `CssClass` per il CSS applicativo circoscritto e i widget di terze parti.
- `CssStyle` solo per valori dinamici una tantum.
- Contenitori di layout per il posizionamento adattivo.
- Proprietà responsive per i valori di profilo impostati nel designer.

- `ResponsiveProfileChanged` per il comportamento a runtime.
- `ClientProfiles.json` per le definizioni dei profili stesse.
- Governance di produzione per una consegna manutenibile.

La griglia di valutazione pesa il design e la coerenza del tema al 25 percento, la correttezza del layout responsive al 25, `ClientProfile` e proprietà responsive al 20, manutenibilità e governance al 15, e spiegazione e allineamento del grounding al 15. Quegli ultimi trenta punti si guadagnano con la nota e il pack: se l'assistente, leggendo solo ciò che hai scritto, sa spiegare perché la regione dei dettagli è una finestra modale sul telefono e perché la card delle metriche è una `CssClass` invece di un'appearance, la console è finita.



La regola del sistema visivo

Usa il meccanismo Wisej.NET che possiede il problema: il tema per l'identità dei controlli standard, `AppearanceKey` per le varianti semantiche, `CssClass` per il CSS circoscritto e specifico dell'applicazione, `CssStyle` solo per valori dinamici una tantum, i contenitori di layout invece del codice di ridimensionamento, le proprietà responsive per i valori di profilo impostati nel designer, e `ResponsiveProfileChanged` per il comportamento che non è semplicemente visivo.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Consegnare una UI brandizzata per desktop, tablet e telefono che usa in modo coerente tema, contenitori di layout e `ClientProfile`.
- ✓ Rivedere l'accessibilità visiva: indicatori di focus, stati non validi, contrasto, aree di tocco, ordine di tabulazione e comportamento dei contenuti nascosti.
- ✓ Preparare una checklist di produzione per file di tema, file CSS, `ClientProfiles.json`, screenshot, test di accettazione e personalizzazioni sicure rispetto agli aggiornamenti.
- ✓ Impacchettare la mappa delle fonti e i materiali RAG così che un assistente IA con grounding possa rispondere a domande di implementazione a partire dal corso.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.



Lab pratico

Completa il capstone di Adaptive Operations Console: un tema personalizzato con token con nome e almeno quattro appearance semantiche, un uso circoscritto di `CssClass`, uno shell desktop con regioni `Flow`, `Table` e `Flex`, `ClientProfiles.json` con comportamento per telefono, tablet, small desktop e desktop, proprietà responsive più un gestore `ResponsiveProfileChanged` idempotente, e un indicatore del profilo visibile in modalità di test. Esegui la revisione di accessibilità per focus frame, stati non validi, contrasto, aree di tocco e ordine di tabulazione, compila la matrice di QA responsive con screenshot per ogni profilo, e scrivi la nota di architettura e il grounding pack che spiegano ogni decisione su tema, layout e profili.

Punto di partenza

Apri la soluzione Adaptive Operations Console che hai portato avanti dal Modulo 1 al Modulo 6: `AdaptiveOps.theme` e il foglio di stile circoscritto `AdaptiveOps.css` esistono, lo shell è fatto di contenitori annidati con regioni `Flow`, `Table` e `Flex`, `ClientProfiles.json` elenca sei profili e il gestore del Modulo 6 mostra già il nome del profilo attivo nella label di stato. In questo lab non si costruisce nulla di nuovo da zero; ogni passo irrobustisce, rivede o documenta ciò che c'è già.

Approccio suggerito

- ✓ Verifica prima i livelli visivi: conferma che il tema derivi da un tema di base con token con nome, conta le appearance semantiche assegnate tramite `AppearanceKey` (aggiungi un `destructive-button`, `kpi-card` o `nav-item` se ne hai meno di quattro), e sposta ogni override isolato di `BackColor`, `ForeColor` o `Font` nel tema o in una `CssClass`.
- ✓ Irrobustisci il layout: imposta `MinimumSize` su ogni regione con `FillWeight` o in dock e `MaximumSize` dove una regione non deve crescere, toglie `AutoSize` dalle grandi gerarchie di contenitori, e ridimensiona attraverso tutte le larghezze per verificare che nulla collassi o vada in overflow orizzontale.
- ✓ Riscrivi la logica dei profili come un unico metodo `ApplyProfile(string name)` che assegna valori finali ed esce subito quando il profilo non è cambiato; chiamalo dal costruttore con `Application.ActiveProfile` e dal gestore `ResponsiveProfileChanged`, annulla la sottoscrizione in `Dispose` e mostra `lblProfile` solo quando è impostato un flag di modalità di test.
- ✓ Esegui la revisione di accessibilità usando davvero la schermata: attraversa ogni regione con `Tab` e correggi `TabIndex` e `TabStop`, attiva la validazione e verifica che lo stato non valido abbia un'icona o un messaggio oltre al colore, controlla il contrasto per ogni coppia di token, aggiungi `ToolTipText` ai pulsanti solo icona e misura le aree di tocco alla larghezza telefono.
- ✓ Compila la matrice di QA responsive (desktop, small desktop, tablet verticale e orizzontale, telefono verticale e orizzontale) con il profilo atteso, uno screenshot e i controlli per ogni riga,

poi scrivi la nota di architettura e il grounding pack: albero decisionale, regole di risposta, mappa delle fonti e una domanda a cui l'assistente deve rispondere a partire dal tuo materiale.

Deliverable richiesti

- ✓ Adaptive Operations Console completa con tema personalizzato, appearance semantiche, CSS circoscritto e regioni Flow, Table e Flex
- ✓ ClientProfiles.json, proprietà responsive e un gestore ResponsiveProfileChanged idempotente con un indicatore del profilo in modalità di test
- ✓ Revisione di accessibilità che copre focus frame, stati non validi, contrasto, aree di tocco e ordine di tabulazione
- ✓ Matrice di QA responsive con screenshot desktop, tablet e telefono in entrambi gli orientamenti
- ✓ Nota di architettura e grounding pack per l'IA che spiegano ogni decisione su tema, layout e profili

Domande di revisione per lo studente

- ✓ Il tuo gestore `ResponsiveProfileChanged` viene eseguito due volte per il profilo Tablet dopo un ricaricamento del browser. Quale tua riga garantisce che la seconda esecuzione non cambi nulla, e che cosa vedrebbe l'utente se quella riga mancasse?
- ✓ Un revisore ti chiede di rimuovere il focus frame da `action-button` perché "sembra datato". Che cosa cambi invece, quale token lo governa, e quali altri due stati ricontrolli poi nel Theme Builder?
- ✓ L'intestazione della griglia dei ticket ha bisogno di un ritocco che solo un selettore sul DOM interno del widget può raggiungere. Che cosa provi per prima cosa, che cosa deve essere vero prima che il selettore sia ammesso, e che cosa metti per iscritto così che il prossimo aggiornamento non sorprenda il team?

Consegna del capstone

Questo lab è anche la consegna del capstone del corso. Consegna il progetto sorgente, AdaptiveOps.theme, AdaptiveOps.css, ClientProfiles.json, la matrice di QA responsive completa con i suoi screenshot, la revisione di accessibilità, la nota di architettura e il grounding pack. Preparati a mostrare la console su ogni profilo e a rispondere, per qualsiasi controllo sullo schermo, quale meccanismo possiede il suo aspetto e perché.

Area di valutazione	Peso	Che cosa significa "eccellente"
Design e coerenza del tema	25%	Tema derivato da una base, token riutilizzati ovunque, appearance semantiche per ogni variante di business, nessun override improvvisato di colori o font.
Correttezza del layout responsive	25%	Il layout lo possiedono i contenitori, le regioni hanno dimensioni minime, ogni larghezza della matrice di QA è utilizzabile in entrambi gli orientamenti.
ClientProfile e proprietà responsive	20%	Ordine dei profili a prima corrispondenza, prima i valori assegnati nel designer, un unico gestore idempotente per i veri cambi di comportamento, indicatore in modalità di test.
Manutenibilità e governance	15%	File nel controllo del codice sorgente, diff revisionati, eccezioni CSS documentate, matrice di screenshot e revisione di accessibilità complete e ripetibili dopo gli aggiornamenti.
Spiegazione e allineamento del grounding per l'IA	15%	Nota di architettura e grounding pack a partire dai quali un assistente può spiegare ogni decisione e mettere in guardia dalle scelte fragili.



La regola del sistema visivo

Usa il meccanismo Wisej.NET che possiede il problema: il tema per l'identità dei controlli standard, AppearanceKey per le varianti semantiche, CssClass per il CSS circoscritto e specifico dell'applicazione, CssStyle solo per valori dinamici una tantum, i contenitori di layout invece del codice di ridimensionamento, le proprietà responsive per i valori di profilo impostati nel designer, e ResponsiveProfileChanged per il comportamento che non è semplicemente visivo.