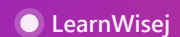


Projet de synthèse prêt pour le mobile, accessibilité, performances et gouvernance en production

Guide de la leçon · Guide du lab / de l'examen



Guide de la leçon

Finaliser Adaptive Operations Console comme un ensemble d'écrans bureau, tablette et téléphone prêt pour la production, contrôler le focus, le contraste, les états invalides et les zones tactiles, garder des gestionnaires de profil légers et idempotents, et gouverner le thème, le CSS et ClientProfiles.json comme du code. Ce que cela signifie pour un développeur Wisej.NET — et comment aborder ce module.



Enjeu en production

Finaliser Adaptive Operations Console comme un ensemble d'écrans bureau, tablette et téléphone prêt pour la production, contrôler le focus, le contraste, les états invalides et les zones tactiles, garder des gestionnaires de profil légers et idempotents, et gouverner le thème, le CSS et ClientProfiles.json comme du code.

1. Le problème en production

Six modules de travail vivent désormais dans une seule solution : un thème dérivé d'un thème de base, du CSS ciblé, un shell de conteneurs imbriqués, des régions Flow, Table et Flex, un ClientProfiles.json et un gestionnaire de profil. Ce qui transforme cela en produit n'est pas une fonctionnalité de plus. C'est la preuve que la console est utilisable sur chaque profil, qu'un utilisateur au clavier ou au toucher peut s'en servir, qu'un redimensionnement ne déclenche pas de reconstructions coûteuses, et que le développeur suivant, ou la prochaine mise à jour de Wisej.NET, ne peut pas casser sans bruit le système visuel. Ce module apporte cette preuve, ainsi que le grounding pack qui permet à un assistant IA de répondre aux questions sur les décisions que vous avez prises.

2. Concepts clés

Concept	Pourquoi c'est important
ensemble d'écrans du projet de synthèse	Adaptive Operations Console terminée : thème personnalisé, appearances sémantiques, CSS ciblé, un shell bureau, un shell tablette adaptatif, un shell téléphone, une grille, un éditeur, un retour de validation, une navigation et un indicateur de profil visible en mode test.
revue d'accessibilité visuelle	Un passage sur les cadres de focus, les états désactivés et invalides, le contraste, l'espacement tactile, les infobulles des boutons à icône seule et l'ordre de tabulation au clavier, réalisé en utilisant l'écran plutôt qu'en lisant le thème.
gestionnaire de profil idempotent	Un gestionnaire <code>ResponsiveProfileChanged</code> qui attribue des valeurs finales pour le profil courant, si bien que l'exécuter deux fois, ou après un rafraîchissement, produit exactement la même interface.
stabilité de la mise en page	<code>MinimumSize</code> et <code>MaximumSize</code> empêchent les régions de s'effondrer, et <code>AutoSize</code> reste désactivé sur les grandes hiérarchies de conteneurs, pour des redimensionnements prévisibles et peu coûteux.
gouvernance du thème	Le JSON du thème, la feuille de style ciblée et ClientProfiles.json sont versionnés, relus sous forme de diffs, documentés en tokens et jamais modifiés à l'intérieur d'un thème intégré.
personnalisation résistante aux mises à jour	Les appearances documentées, les flux du Theme Builder et les propriétés côté serveur survivent aux mises à jour ; les sélecteurs CSS qui dépendent du DOM interne d'un widget peut-être pas, ils sont donc ciblés, documentés ou évités.
matrice de QA responsive et grounding pack	La matrice consigne les largeurs, orientations, profil attendu et vérifications de chaque scénario ; le grounding pack consigne l'arbre de décision, les règles de réponse et la carte des sources à partir desquels un assistant répond.

3. Déroulé de la leçon

Une console, trois shells, un système visuel

Le projet de synthèse est évalué sur la cohérence, pas sur la nouveauté. Les contrôles standard tirent leur identité du thème AdaptiveOps, qui dérive d'un thème de base et définit des tokens de couleurs nommés pour la marque, les surfaces, le texte, danger, warning, success et le cadre de focus. Les variantes métier sont des appearances sémantiques attribuées via `AppearanceKey` : `action-button`, `destructive-button`, `kpi-card`, `nav-item`, `compact-editor` ou équivalents, au moins quatre. Le contenu propre à l'application, comme les cartes de métriques, utilise des classes ciblées via `CssClass`, et `CssStyle` n'apparaît que là où une valeur est calculée à l'exécution. Si un écran recourt directement à `BackColor`, `ForeColor` ou `Font`, la revue demande laquelle de ces couches aurait dû porter la décision.

Le shell conserve la structure construite aux Modules 4 et 5 : barre d'outils ancrée, rail de navigation, espace de travail, région de détails et pied de page d'état, avec un `FlowLayoutPanel` pour les filtres et les cartes, un `TableLayoutPanel` pour l'éditeur de ticket aligné et un `FlexLayoutPanel` pour l'espace de

travail proportionnel. Les différences entre bureau, tablette et téléphone viennent de ClientProfiles.json et des propriétés responsives attribuées dans le designer ; l'unique gestionnaire `ResponsiveProfileChanged` existe pour le comportement qu'une propriété ne peut pas exprimer, comme ouvrir l'éditeur de détails sous forme de formulaire modal sur téléphone. L'indicateur de profil reste dans la région d'état, visible en mode test, pour que chaque capture de la matrice de QA nomme le profil qui l'a produite.

Une accessibilité vérifiable en utilisant l'écran

Un thème doit préserver un focus visible, des états désactivés clairs, un texte lisible, un contraste suffisant, un retour sur les états invalides et un espacement adapté au toucher. La régression la plus courante consiste à supprimer le cadre de focus parce qu'il paraît daté ; la règle est de le remplacer par un meilleur, basé sur le token `focusFrame`, jamais de le retirer. Les états invalid et warning ne doivent pas reposer uniquement sur la couleur : associez la bordure danger à une icône ou à un message lisible, pour que l'état reste perceptible malgré le daltonisme ou un vidéoprojecteur. Le contraste se vérifie sur les paires de tokens réellement utilisées par le thème, par exemple `textMuted` sur `surfaceAlt`, pas sur la couleur de la marque prise isolément.

Les boutons de barre d'outils à icône seule, vers lesquels bascule le profil tablette via la valeur responsive `Display`, ont besoin de `ToolTipText` ou d'un libellé accessible pour rester identifiables. L'ordre de tabulation se vérifie en parcourant l'écran au clavier et en confirmant que `TabStop` et `TabIndex` placent la commande principale là où l'utilisateur l'attend. Enfin, le contenu masqué a besoin d'une solution de repli : quand le rail de navigation et le panneau de détails sont masqués sur téléphone, les fonctions qu'ils portaient doivent rester accessibles, via le bouton du menu mobile et l'éditeur modal, et la note d'architecture consigne cette alternative pour chaque contrôle masqué.

Gestionnaires légers, mises en page stables, fichiers gouvernés

Le responsive devient coûteux quand chaque redimensionnement provoque une mise en page complexe, des rechargements de données et des mises à jour du client. Le gestionnaire qui réagit à un changement de profil doit être idempotent et léger : il attribue la visibilité, le docking et la taille finaux pour le profil courant au lieu de les basculer, il ne reconstruit pas l'interface quand quelques changements de propriétés suffisent, et il ne recharge jamais les données simplement parce que la largeur a changé. La méthode `ApplyProfile` du lab est appelée une fois depuis le constructeur avec `Application.ActiveProfile`, puis depuis le gestionnaire avec le profil courant de l'événement, et l'abonnement est retiré lorsque la page est supprimée :

```
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 stabilité de la mise en page découle de la même discipline : `MinimumSize` et `MaximumSize` sur les régions pondérées et ancrées pour que rien ne s'effondre à zéro, pas d' `AutoSize` sur les grandes hiérarchies de conteneurs, et des changements de données responsives intentionnels plutôt qu'un effet de bord d'un redimensionnement. Traitez ensuite le thème comme du code. `AdaptiveOps.theme`, `AdaptiveOps.css` et `ClientProfiles.json` vivent dans le contrôle de source avec des noms clairs ; leurs diffs sont relus ; les tokens de design sont documentés ; les thèmes intégrés ne sont jamais modifiés directement. L'équipe consigne par écrit quand un développeur peut utiliser `BackColor` , `ForeColor` ou `Font` , quand `AppearanceKey` , `CssClass` , `CssStyle` ou du CSS personnalisé est le bon outil, et conserve un jeu de captures des principaux écrans aux tailles bureau, tablette et téléphone.

La résistance aux mises à jour découle des mêmes règles. Les appearances documentées du thème, les flux du Theme Builder et les propriétés côté serveur sont stables d'une version de Wisej.NET à l'autre ; un sélecteur CSS qui dépend du DOM interne d'un widget ne l'est pas. Quand un widget tiers ou personnalisé a réellement besoin de CSS direct, ciblez étroitement le sélecteur et documentez pourquoi le système de thème ne porte pas cette partie. Après chaque mise à jour, relancez la matrice de captures responsive et inspectez chaque clé d'apparence personnalisée et chaque état propre à un widget.



Pièges courants

Supprimer les cadres de focus au lieu de les restyler. Signaler un état invalide uniquement par la couleur. Un gestionnaire de profil qui bascule l'état ou reconstruit toute la page à chaque changement. Des régions pondérées sans `MinimumSize`. Modifier un thème intégré sur place. Du CSS global plaqué sur les contrôles standard via des sélecteurs du DOM interne. Un jeu de captures réalisé une fois et jamais relancé après une mise à jour.

Un grounding pack à partir duquel un assistant peut répondre

Le dernier livrable s'adresse à deux lecteurs : le développeur suivant et un assistant IA ancré. La note d'architecture explique chaque décision de thème, de mise en page et de profil, et quel mécanisme la porte. Le grounding pack ajoute les règles de réponse que l'assistant doit suivre : répondre d'abord à partir du cours, puis à partir de la documentation officielle de la carte des sources ; mettre en garde contre les sélecteurs internes fragiles et contre la modification d'un thème partagé pour toutes les sessions quand une seule session était visée ; ne jamais proposer les media queries CSS comme mécanisme responsive principal quand les propriétés responsives ou `ClientProfile` conviennent ; et demander le profil et la taille d'écran quand un problème responsive est ambigu. La carte des sources liste les pages de documentation derrière chaque règle, pour que l'assistant puisse citer plutôt que deviner.

Au cœur du pack se trouve l'arbre de décision du cours, appliqué dans l'ordre jusqu'à ce qu'un échelon convienne :

- D'abord le thème, pour l'identité des contrôles standard.
- `AppearanceKey` pour les variantes sémantiques.
- `CssClass` pour le CSS applicatif ciblé et les widgets tiers.
- `CssStyle` uniquement pour les valeurs dynamiques ponctuelles.
- Les conteneurs de mise en page pour le positionnement adaptatif.
- Les propriétés responsives pour les valeurs de profil définies dans le designer.

- `ResponsiveProfileChanged` pour le comportement à l'exécution.
- `ClientProfiles.json` pour les définitions de profils elles-mêmes.
- La gouvernance de production pour une livraison maintenable.

La grille d'évaluation pondère la conception et la cohérence du thème à 25 %, la justesse de la mise en page responsive à 25 %, `ClientProfile` et les propriétés responsives à 20 %, la maintenabilité et la gouvernance à 15 %, et l'explication et l'alignement du `grounding` à 15 %. Ces trente derniers points se gagnent avec la note et le pack : si l'assistant, en lisant uniquement ce que vous avez écrit, peut expliquer pourquoi la région de détails est un formulaire modal sur téléphone et pourquoi la carte de métrique est une `CssClass` plutôt qu'une `appearance`, la console est terminée.



Réflexe du système visuel

Utilisez le mécanisme `Wisej.NET` qui porte le problème : le thème pour l'identité des contrôles standard, `AppearanceKey` pour les variantes sémantiques, `CssClass` pour le CSS ciblé propre à l'application, `CssStyle` uniquement pour les valeurs dynamiques ponctuelles, les conteneurs de mise en page plutôt que du code de redimensionnement, les propriétés responsives pour les valeurs de profil définies dans le designer, et `ResponsiveProfileChanged` pour le comportement qui n'est pas simplement visuel.



Objectifs d'apprentissage

À la fin de ce module, vous serez capable de :

- ✓ Livrer une interface bureau/tablette/téléphone à l'image de la marque qui utilise le thème, les conteneurs de mise en page et `ClientProfile` de façon cohérente.
- ✓ Mener une revue d'accessibilité visuelle : indicateurs de focus, états invalides, contraste, zones tactiles, ordre de tabulation et comportement du contenu masqué.
- ✓ Préparer une checklist de production pour les fichiers de thème, les fichiers CSS, `ClientProfiles.json`, les captures d'écran, les tests d'acceptation et les personnalisations résistantes aux mises à jour.
- ✓ Assembler la carte des sources et les supports RAG pour qu'un assistant IA ancré puisse répondre aux questions d'implémentation à partir du cours.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, l'approche suggérée et les livrables attendus.



Lab pratique

Terminez le projet de synthèse Adaptive Operations Console : un thème personnalisé avec des tokens nommés et au moins quatre appearances sémantiques, un usage ciblé de `CssClass`, un shell de bureau avec des régions `Flow`, `Table` et `Flex`, un `ClientProfiles.json` avec un comportement téléphone, tablette, petit bureau et bureau, des propriétés responsives ainsi qu'un gestionnaire `ResponsiveProfileChanged` idempotent, et un indicateur de profil visible en mode test. Menez la revue d'accessibilité des cadres de focus, des états invalides, du contraste, des zones tactiles et de l'ordre de tabulation, remplissez la matrice de QA responsive avec des captures pour chaque profil, et rédigez la note d'architecture et le grounding pack qui expliquent chaque décision de thème, de mise en page et de profil.

Point de départ

Ouvrez la solution Adaptive Operations Console que vous avez fait évoluer des Modules 1 à 6 : `AdaptiveOps.theme` et la feuille de style ciblée `AdaptiveOps.css` existent, le shell est composé de conteneurs imbriqués avec des régions `Flow`, `Table` et `Flex`, `ClientProfiles.json` liste six profils et le gestionnaire du Module 6 affiche déjà le nom du profil actif dans le label d'état. Rien de nouveau n'est construit de zéro dans ce lab ; chaque étape consolide, passe en revue ou documente l'existant.

Approche suggérée

- ✓ Auditez d'abord les couches visuelles : vérifiez que le thème dérive d'un thème de base avec des tokens nommés, comptez les appearances sémantiques attribuées via `AppearanceKey` (ajoutez un `destructive-button`, un `kpi-card` ou un `nav-item` si vous en avez moins de quatre), et déplacez toute redéfinition isolée de `BackColor`, `ForeColor` ou `Font` dans le thème ou dans une `CssClass`.
- ✓ Consolidez la mise en page : définissez `MinimumSize` sur chaque région pondérée ou ancrée et `MaximumSize` là où une région ne doit pas grandir, retirez `AutoSize` des grandes hiérarchies de conteneurs, et redimensionnez à travers toutes les largeurs pour vérifier que rien ne s'effondre ni ne déborde horizontalement.
- ✓ Réécrivez la logique de profil en une seule méthode `ApplyProfile(string name)` qui attribue des valeurs finales et sort immédiatement quand le profil n'a pas changé ; appelez-la depuis le constructeur avec `Application.ActiveProfile` et depuis le gestionnaire `ResponsiveProfileChanged`, désabonnez-vous dans `Dispose`, et n'affichez `lblProfile` que lorsqu'un indicateur de mode test est activé.
- ✓ Menez la revue d'accessibilité en utilisant réellement l'écran : parcourez chaque région avec Tab et corrigez `TabIndex` et `TabStop`, déclenchez la validation et vérifiez que l'état invalide a une icône ou un message en plus de la couleur, contrôlez le contraste de chaque paire de tokens, ajoutez `ToolTipText` aux boutons à icône seule, et mesurez les zones tactiles à la largeur téléphone.

- ✓ Remplissez la matrice de QA responsive (bureau, petit bureau, tablette portrait et paysage, téléphone portrait et paysage) avec le profil attendu, une capture et les vérifications de chaque ligne, puis rédigez la note d'architecture et le grounding pack : arbre de décision, règles de réponse, carte des sources et une question à laquelle l'assistant doit répondre à partir de votre matériel.

Livrables attendus

- ✓ Adaptive Operations Console terminée avec thème personnalisé, appearances sémantiques, CSS ciblé et régions Flow, Table et Flex
- ✓ ClientProfiles.json, propriétés responsives et gestionnaire ResponsiveProfileChanged idempotent avec un indicateur de profil en mode test
- ✓ Revue d'accessibilité couvrant les cadres de focus, les états invalides, le contraste, les zones tactiles et l'ordre de tabulation
- ✓ Matrice de QA responsive avec des captures bureau, tablette et téléphone dans les deux orientations
- ✓ Note d'architecture et grounding pack pour l'IA expliquant chaque décision de thème, de mise en page et de profil

Questions de révision

- ✓ Votre gestionnaire `ResponsiveProfileChanged` s'exécute deux fois pour le profil Tablet après un rafraîchissement du navigateur. Quelle ligne de votre code garantit que la seconde exécution ne change rien, et que verrait l'utilisateur si cette ligne manquait ?
- ✓ Un relecteur vous demande de retirer le cadre de focus d' `action-button` parce qu'il « fait daté ». Que modifiez-vous à la place, quel token le pilote, et quels deux autres états revérifiez-vous ensuite dans le Theme Builder ?
- ✓ L'en-tête de la grille des tickets a besoin d'un ajustement que seul un sélecteur sur le DOM interne du widget peut atteindre. Qu'essayez-vous en premier, qu'est-ce qui doit être vrai avant que le sélecteur soit autorisé, et que consignez-vous pour que la prochaine mise à jour ne surprenne pas l'équipe ?

Remise du projet de synthèse

Ce lab constitue aussi la remise du projet de synthèse du cours. Remettez le projet source, AdaptiveOps.theme, AdaptiveOps.css, ClientProfiles.json, la matrice de QA responsive remplie avec ses captures, la revue d'accessibilité, la note d'architecture et le grounding pack. Soyez prêt à parcourir la console sur chaque profil et à expliquer, pour n'importe quel contrôle à l'écran, quel mécanisme porte son apparence et pourquoi.

Critère d'évaluation	Poids	À quoi ressemble l'excellence
Conception et cohérence du thème	25 %	Thème dérivé d'une base, tokens réutilisés partout, appearances sémantiques pour chaque variante métier, aucune redéfinition improvisée de couleur ou de police.
Justesse de la mise en page responsive	25 %	Les conteneurs portent la mise en page, les régions ont des tailles minimales, chaque largeur de la matrice de QA est utilisable dans les deux orientations.
ClientProfile et propriétés responsives	20 %	Ordre des profils par première correspondance, valeurs attribuées d'abord dans le designer, un seul gestionnaire idempotent pour les vrais changements de comportement, indicateur en mode test.
Maintenabilité et gouvernance	15 %	Fichiers dans le contrôle de source, diffs relus, exceptions CSS documentées, matrice de captures et revue d'accessibilité complètes et reproductibles après les mises à jour.
Explication et alignement du grounding pour l'IA	15 %	Note d'architecture et grounding pack à partir desquels un assistant peut expliquer chaque décision et mettre en garde contre les choix fragiles.



Réflexe du système visuel

Utilisez le mécanisme Wisej.NET qui porte le problème : le thème pour l'identité des contrôles standard, AppearanceKey pour les variantes sémantiques, CssClass pour le CSS ciblé propre à l'application, CssStyle uniquement pour les valeurs dynamiques ponctuelles, les conteneurs de mise en page plutôt que du code de redimensionnement, les propriétés responsives pour les valeurs de profil définies dans le designer, et ResponsiveProfileChanged pour le comportement qui n'est pas simplement visuel.