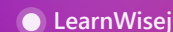


# Shells mobiles avec .NET MAUI

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



## Guide de la leçon

Héberger l'application dans un shell hybride .NET MAUI pour Android et iOS, navigation et comportement du bouton Retour, cycle de vie de l'application et particularités des plateformes. Ce que cela signifie pour une équipe qui livre une application Wisej.NET au-delà de l'onglet du navigateur — et comment aborder ce module.



### Enjeu en production

Héberger l'application dans un shell hybride .NET MAUI pour Android et iOS, navigation et comportement du bouton Retour, cycle de vie de l'application et particularités des plateformes.

## 1. Le problème en production

Les techniciens de FieldOps travaillent depuis un téléphone dans une salle des machines, un sous-sol ou une camionnette, et un onglet de navigateur est un mauvais conteneur pour cela : pas d'icône d'écran d'accueil à distribuer, pas de canal store ou entreprise, et aucun accès à la caméra et au GPS dont le Module 5 aura besoin. Le système d'exploitation mobile est aussi un hôte bien plus agressif — il suspend l'application dès qu'un appel arrive, la redessine sous une encoche et offre au technicien un geste Retour système que FieldOps n'a jamais demandé.

## 2. Concepts clés

| Concept                             | Pourquoi c'est important                                                                                                                                                                                                        |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| shell hybride .NET MAUI             | Une petite application native pour Android et iOS dont l'unique page héberge votre application Wisej.NET dans une web view, de sorte que les écrans C# que vous livrez déjà sont ce que voit le technicien.                     |
| hôte web view natif                 | Le moteur de navigateur de la plateforme intégré à cette page — Android System WebView, WKWebView sur iOS — qui exécute le même client qu'un navigateur desktop, ce qui explique qu'aucun écran ne soit réécrit pour le mobile. |
| bridge hybride                      | Le canal bidirectionnel entre la page hébergée et le code natif, qui permet au C# côté serveur de demander quelque chose à l'appareil et d'en recevoir la réponse sous forme d'événement plutôt que de navigation de page.      |
| cycle de vie d'une application MAUI | Les événements que le système lève autour de l'application — fenêtre créée, activée, désactivée, arrêtée et reprise, plus <code>OnSleep</code> et <code>OnResume</code> — dont aucun n'est levé par un onglet de navigateur.    |
| fenêtre de survie de la session     | La durée passée en arrière-plan pendant laquelle la session Wisej.NET est encore active sur le serveur, de sorte qu'une reprise peut se reconnecter à l'état existant au lieu de redémarrer l'application.                      |
| marges de zone sûre                 | Le padding propre à chaque appareil qui éloigne le contenu de l'encoche, de la barre d'état et de la bande de l'indicateur d'accueil, indiqué par la plateforme plutôt que deviné à partir de la taille de l'écran.             |
| gestion de la navigation Retour     | La règle qui décide de l'effet du geste Retour d'Android : revenir en arrière dans la navigation propre à l'application, ou confirmer la sortie — jamais détruire en silence un ordre de travail à moitié rempli.               |

## 3. Déroulé de la leçon

### Un projet de shell, deux stores, le même serveur

Le shell mobile de FieldOps est une application .NET MAUI avec une seule mission : créer une fenêtre, héberger une web view, la pointer vers le serveur FieldOps. Le support de Wisej.NET Hybrid fournit ce shell sous la forme d'un projet que vous construisez et ajustez plutôt que de l'écrire de zéro, et la page qu'il héberge est l'application FieldOps ordinaire du Module 1. Rien ne change pour `Wisej.Web`, le designer de Visual Studio ou `Application.Session` ; les contrôles vivent toujours sur le serveur et la web view affiche toujours leur reflet.

Un seul projet MAUI cible à la fois Android et iOS, et ses dossiers de plateforme ne contiennent que ce qui diffère réellement : l'activité et le manifeste Android, le `Info.plist` et les entitlements iOS, les icônes et les écrans de démarrage. Gardez l'adresse du serveur dans une seule valeur de configuration lue par le shell au démarrage, pour que le débogage pointe vers un portable, le test vers la préproduction et la release vers la production sans modifier le code.

Cette séparation fait aussi l'économie du shell. FieldOps livre du nouveau code serveur dès que vous le déployez ; le shell ne doit être reconstruit que lorsque le shell lui-même change — une nouvelle icône, une nouvelle autorisation, une nouvelle capacité du bridge. Tout ce qui peut vivre sur le serveur doit y vivre, car le serveur est la partie que vous pouvez corriger cet après-midi, alors que la file de revue d'un store se mesure en jours.

### L'arrière-plan est le cas normal, pas le cas d'échec

Sur un téléphone, l'application est interrompue en permanence : un appel, une référence de pièce vérifiée dans une autre application, un écran verrouillé en montant à une échelle. Chaque interruption déclenche la séquence du cycle de vie MAUI — la fenêtre désactivée, puis arrêtée, puis reprise au retour, avec les overrides `OnSleep` et `OnResume` au niveau de l'application qui se déclenchent en parallèle. Gérez-les explicitement, car le comportement par défaut d'une web view évincée de la mémoire est de recharger la page, et un rechargement est une nouvelle session Wisej.NET : le relevé de compteur à moitié saisi a disparu.

La conception qui fonctionne traite un court passage en arrière-plan comme une pause et un long comme un redémarrage. À l'arrêt, enregistrez l'heure et laissez la session inactive. À la reprise, comparez le temps écoulé au délai d'expiration de session configuré sur le serveur FieldOps ; si la session doit encore être active, reconnectez-vous et laissez le serveur repousser l'état courant plutôt que de faire naviguer à nouveau la web view. Si l'interruption dépasse le délai, ou si la plateforme indique que l'application a été terminée faute de mémoire, restaurez délibérément : rechargez, puis rouvrez l'ordre de travail à l'aide d'un identifiant que le shell a conservé localement. Traitez un avertissement de mémoire faible exactement comme un arrêt.

```
protected override Window CreateWindow(IActivationState state)
{
    var window = base.CreateWindow(state);
    window.Stopped += (s, e) => ShellState.Backgrounded(DateTime.UtcNow);
    window.Resumed += (s, e) => ShellState.Resume(); // reconnect, don't reload
    return window;
}

protected override void OnSleep() => ShellState.Backgrounded(DateTime.UtcNow);
protected override void OnResume() => ShellState.Resume();
```

Le côté serveur du contrat compte tout autant : gardez un délai d'expiration de session FieldOps assez long pour couvrir une interruption réaliste, et faites en sorte que l'éditeur d'ordre de travail enregistre un brouillon à chaque modification de champ plutôt qu'uniquement à la validation.

### Retour, encoches et la bande du bas

Le geste Retour d'Android est un événement système, et l'ignorer revient à appliquer le comportement par défaut de la plateforme : l'activité se ferme, si bien qu'un technicien qui fait un geste Retour depuis une page de détail d'ordre de travail perd l'application au lieu de revenir à la liste. Interceptez la navigation Retour dans le shell et acheminez-la d'abord vers l'application web — la page de détail revient à la liste, une boîte de dialogue ouverte se ferme — et ne passez à une demande de confirmation que lorsqu'il ne reste plus rien à dépiler. L'état qui répond à la question « y a-t-il quelque chose vers quoi revenir » vit sur

le serveur ; le shell doit donc interroger l'application plutôt que deviner : gardez une petite pile de navigation explicite dans FieldOps lui-même. iOS n'a pas de bouton Retour, donc la commande Retour propre à l'application doit exister dans tous les cas.

Les zones sûres sont l'autre particularité visible des plateformes. Un téléphone à encoche réserve la bande du haut et un iPhone récent réserve en bas une bande pour l'indicateur d'accueil ; le contenu dessiné à ces endroits est masqué ou impossible à toucher. Lisez les marges indiquées par la plateforme et appliquez-les comme padding autour du contenu hébergé, pour que la barre d'application de FieldOps se trouve sous la barre d'état et que les boutons de mise à jour du statut se trouvent au-dessus de l'indicateur d'accueil. Appliquez le padding à l'hôte plutôt que d'écrire des valeurs fixes en pixels dans la mise en page Wisej.NET, et les mêmes écrans restent corrects dans un simple navigateur, sur un appareil sans encoche et en mode paysage.

## Émulateurs, appareils et deux commandes de build

La boucle de build est propre à chaque plateforme et mérite d'être mise en place avant d'écrire le moindre code de shell. Android a besoin du SDK et d'au moins une image d'Android Emulator ; une exécution de débogage se fait avec `dotnet build -t:Run -f net8.0-android` ou la cible équivalente dans Visual Studio. iOS a besoin d'un Mac dans la boucle — l'iOS Simulator et un build comme `dotnet build -f net8.0-ios` — même lorsque vous développez sous Windows et vous appelez à distance à ce Mac. Le hot restart raccourcit le cycle, et comme la plupart des modifications de FieldOps sont côté serveur, beaucoup ne nécessitent aucun rebuild : modifiez le C#, redémarrez le serveur, tirez pour actualiser dans le shell.

Le débogage couvre deux runtimes : les points d'arrêt dans le C# du shell couvrent le cycle de vie et les appels au bridge, ceux du code serveur couvrent les écrans. Lorsque l'émulateur n'arrive pas à joindre votre serveur de développement, la cause est presque toujours l'adresse loopback — le `localhost` d'un émulateur est l'appareil émulé, pas votre machine — ou un certificat auto-signé que la plateforme refuse. Validez ensuite le shell sur du vrai matériel : les zones tactiles, le clavier qui recouvre un champ et l'éviction sous pression mémoire s'y comportent tous différemment, et les captures d'écran des deux émulateurs réalisées dans le lab sont la référence avec laquelle comparer cette exécution sur appareil.



### Réflexe de packaging

L'application reste une application web Wisej.NET servie par votre serveur ; le shell est un véhicule de distribution. Gardez chaque fonctionnalité opérationnelle dans un simple navigateur, puis ajoutez les capacités de l'appareil comme améliorations progressives derrière une vérification de capacités.



### Objectifs d'apprentissage

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

- ✓ Héberger l'application Wisej.NET dans un shell hybride .NET MAUI ciblant Android et iOS.
- ✓ Gérer les événements du cycle de vie mobile — arrière-plan, reprise, mémoire faible — sans perdre la session.
- ✓ Associer la navigation de la plateforme, le bouton Retour et les zones sûres à l'application web.

- ✓ Mettre en place les émulateurs, le débogage sur appareil et la chaîne de build pour les deux plateformes.

## Guide du lab / de l'examen

---

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



### Lab pratique

Construisez le shell mobile de FieldOps avec .NET MAUI : une page hybride qui héberge l'application sur Android et iOS, une session qui survit au passage en arrière-plan et reprend au retour, le bouton Retour d'Android associé à la navigation dans l'application, des marges de zone sûre pour les appareils à encoche, et un build de débogage exécuté sur un émulateur par plateforme avec des captures d'écran de la liste des ordres de travail.

### Point de départ

Vous ouvrez la solution FieldOps du Module 1 — la liste des ordres de travail, la page de détail et le flux de mise à jour du statut, qui fonctionnent déjà dans un simple navigateur et, depuis le Module 2, s'installent comme PWA. Rien ne change dans ces écrans pendant ce lab ; vous ajoutez une seconde tête à côté du projet web, un shell .NET MAUI qui héberge la même application rendue par le serveur sur un téléphone.

### Approche suggérée

- ✓ Ajoutez le projet de shell MAUI à la solution FieldOps, ciblez Android et iOS, et lisez l'adresse du serveur depuis la configuration pour que les builds de débogage et de release pointent vers des hôtes différents.
- ✓ Hébergez l'application FieldOps dans la web view du shell et vérifiez que la liste des ordres de travail, la page de détail et une mise à jour de statut fonctionnent de bout en bout avant de toucher à quoi que ce soit de spécifique à une plateforme.
- ✓ Gérez les événements d'arrêt, de reprise et de mémoire faible de la fenêtre : enregistrez l'heure du passage en arrière-plan et, lors d'une reprise dans le délai d'expiration de session, reconnectez-vous au lieu de recharger la page.
- ✓ Interceptez la navigation Retour d'Android et acheminez-la vers la pile de navigation de FieldOps, en ne demandant confirmation que lorsqu'il ne reste plus rien vers quoi revenir ; appliquez les marges de zone sûre indiquées comme padding autour du contenu hébergé.
- ✓ Exécutez un build de débogage sur l'Android Emulator et sur l'iOS Simulator, capturez la liste des ordres de travail sur chacun, et notez chaque différence dont vous avez dû tenir compte selon la plateforme.

### Livrables attendus

- ✓ Projet de shell hybride .NET MAUI pour Android et iOS
- ✓ Gestion du cycle de vie pour l'arrière-plan et la reprise
- ✓ Gestion du bouton Retour et des zones sûres
- ✓ Builds sur émulateur pour les deux plateformes
- ✓ Captures d'écran des deux émulateurs

## Questions de révision

- ✓ Un technicien est à mi-chemin d'une mise à jour de statut lorsqu'un appel de trois minutes arrive. Qu'est-ce qui doit être vrai sur le serveur et dans le shell pour que la mise à jour inachevée soit toujours là ensuite, et qu'est-ce qui change si l'appel dure une heure ?
- ✓ Le shell ne sait rien de l'écran FieldOps ouvert. Comment un geste Retour décide-t-il entre revenir de la page de détail, fermer une boîte de dialogue et quitter l'application, et où cet état se trouve-t-il réellement ?
- ✓ Lesquelles des modifications de ce module un technicien remarquerait-il encore s'il ouvrait FieldOps dans le navigateur du téléphone plutôt que dans le shell, et lesquelles disparaîtraient ? Qu'est-ce que cela vous apprend sur l'endroit où placer le padding de zone sûre ?



### Réflexe de packaging

L'application reste une application web Wisej.NET servie par votre serveur ; le shell est un véhicule de distribution. Gardez chaque fonctionnalité opérationnelle dans un simple navigateur, puis ajoutez les capacités de l'appareil comme améliorations progressives derrière une vérification de capacités.