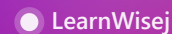


API de l'appareil via le bridge hybride

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



Guide de la leçon

Caméra, fichiers, géolocalisation, notifications push et biométrie depuis l'application via le shell, avec vérifications de capacités et demandes d'autorisation. 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

Caméra, fichiers, géolocalisation, notifications push et biométrie depuis l'application via le shell, avec vérifications de capacités et demandes d'autorisation.

1. Le problème en production

Un technicien FieldOps photographie une vanne corrodée, recueille une signature sur l'écran et part vers l'adresse suivante — pendant que le répartiteur qui a attribué l'intervention suit le même ordre de travail dans un navigateur desktop. Chaque contrôle de FieldOps est un objet C# sur votre serveur, qui ne peut toucher ni la caméra, ni la puce GPS, ni la zone de notifications. Ce module est l'aller-retour qui comble cet écart, et ce que fait l'écran là où cet aller-retour n'existe pas.

2. Concepts clés

Concept	Pourquoi c'est important
bridge vers l'appareil	Le canal de messages qu'un shell hybride expose à la page qu'il héberge : votre serveur demande au shell d'effectuer une opération sur l'appareil, et le shell répond par un message que l'application retransforme en événement côté serveur.
objet hôte	L'objet qu'une WebView injecte dans la page hébergée pour que le JavaScript de cette page puisse appeler du code natif. Le C# côté serveur ne l'atteint qu'à travers l'interopérabilité JavaScript de l'application.
descripteur de capacités	Les fonctionnalités qu'un shell annonce au démarrage d'une session — caméra, localisation, notifications, biométrie — mises en cache et consultées avant d'afficher toute action sur l'appareil.
amélioration progressive	Concevoir chaque écran pour que le chemin du simple navigateur soit une fonctionnalité complète et que le chemin natif en soit une meilleure version, jamais le seul qui fonctionne.
autorisation au moment opportun	Demander à la plateforme l'accès à la caméra, à la localisation ou aux notifications au moment où l'utilisateur appuie sur le bouton qui en a besoin, juste après une phrase qui explique pourquoi.
événement de résultat du bridge	La réponse asynchrone — octets de la photo, coordonnées ou refus — qui arrive sur une requête ultérieure à celle du clic qui l'a demandée, et qui a donc besoin de son propre push.
stockage hors de la racine web	Écrire les photos et signatures capturées là où aucune URL ne mène, puis les servir via un gestionnaire qui vérifie si cette session a le droit de voir cet ordre de travail.

3. Déroulé de la leçon

Interroger le shell, et attendre la réponse

Le bouton Take Photo n'ouvre pas de caméra. Son gestionnaire compose une requête — une photo fixe, 1600 pixels sur le plus grand côté, qualité JPEG 70 — et la transmet en deux sauts. Le premier est celui que Wisej.NET vous offre déjà : le code côté serveur exécute du JavaScript dans la page hébergée et récupère une valeur, le mécanisme qui sous-tend `Application.Call` et la couche d'interopérabilité des widgets. Le second est celui du shell : ce JavaScript appelle l'objet hôte injecté par la WebView, et le shell invoque la caméra de la plateforme. La réponse revient par le même chemin, levée comme un événement .NET ordinaire portant un tableau d'octets.

Ce qui distingue cela d'un clic normal, c'est le temps. Le technicien peut passer quarante secondes à cadrer la prise ; le gestionnaire ne peut donc pas bloquer : il rend la main immédiatement et la requête reste en attente dans `Application.Session` sous un jeton que vous avez généré. La réponse arrive sur une autre requête ; le code qui enregistre les octets et remplit l'image s'exécute donc en dehors de l'événement d'origine, et ses modifications de contrôles n'atteindront pas le navigateur d'elles-mêmes — terminez par `Application.Update`, exactement comme pour un travail lancé avec `Application.StartTask`.

L'annulation et l'échec sont aussi des résultats, pas des exceptions : l'utilisateur a renoncé, la requête a expiré, le shell a été suspendu puis repris. Tous arrivent par le même événement, et c'est le jeton qui vous permet d'écarter une réponse que plus personne n'attend.

```
private void btnPhoto_Click(object sender, EventArgs e)
{
    if (!Shell.Current.Has("camera")) { uploadPhoto.Visible = true; return; }
    var token = Shell.Current.RequestPhoto(maxEdge: 1600, quality: 70);
    this.pendingPhotos[token] = this.workOrder.Id;
}

private void OnPhotoCaptured(object sender, ShellResultEventArgs e)
{
    if (!this.pendingPhotos.TryGetValue(e.Token, out var id)) return;
    this.photoStore.Save(id, e.Bytes);
    this.picPhoto.Image = Image.FromStream(new MemoryStream(e.Bytes));
    Application.Update(this);
}
```

Un écran, quatre shells

FieldOps est livré dans un navigateur, une PWA, une fenêtre desktop WebView2 et un shell mobile .NET MAUI, et la page d'ordre de travail est la même page dans les quatre. Ce qu'elle affiche provient d'une vérification de capacités effectuée une fois par session : l'application demande à la page si un objet hôte est présent et ce que le shell déclare savoir faire, et cette réponse est placée dans la session pour que chaque écran la lise. Ce qu'il ne faut pas faire, c'est analyser le user agent — il décrit le navigateur, pas le fait que ce build de votre shell expose une caméra.

La matrice comporte une ligne par capacité et une colonne par shell. La caméra est native dans le shell mobile et, dans un navigateur de téléphone, passe par le contrôle **Upload**, que le navigateur alimente directement depuis l'application appareil photo ; le shell desktop propose à la place une boîte de dialogue de fichiers. La localisation est native sur mobile et passe ailleurs par l'API Geolocation du navigateur, atteinte par interopérabilité JavaScript. Les notifications viennent du système d'exploitation dans les deux shells et du web push dans le navigateur et la PWA. La biométrie n'existe que là où la plateforme dispose d'un authenticateur ; FieldOps l'utilise donc pour déverrouiller à nouveau une session, jamais comme seul moyen d'entrer.

Take Photo et **Upload** sont donc deux rendus d'une même action, et le reste de l'écran ne sait jamais lequel s'est exécuté. Une capacité que le shell n'annonce pas n'apparaît tout simplement pas : un bouton désactivé sans explication est pire qu'un bouton absent, et un bouton qui lève une exception quand on l'actionne est pire que les deux.

L'autorisation au moment de l'usage

Deux barrières se dressent devant chaque fonctionnalité de l'appareil : l'autorisation de la plateforme, lorsque le système ou le navigateur demande à l'utilisateur d'autoriser la caméra, la localisation ou les notifications, et votre propre explication, qui vient en premier. FieldOps affiche une phrase — la localisation indique la distance jusqu'au prochain site et horodate l'heure d'arrivée — et seulement ensuite déclenche la demande de la plateforme.

Le moment décide de la réponse. Une demande de localisation sur l'écran de connexion, avant que le technicien n'ait vu un ordre de travail, reçoit un « non », et sur plusieurs plateformes ce non est définitif ; la même demande déclenchée par Navigate sur une intervention dont l'adresse est à l'écran reçoit un « oui » presque à chaque fois. FieldOps demande donc les notifications après la première attribution de la session, et la caméra lorsque Take Photo est pressé.

- Accordée : continuez, et mémorisez l'accord pour cette session.
- Refusée une fois : conservez l'action et la raison ; la prochaine intervention obtiendra peut-être une autre réponse.
- Refusée définitivement : substituez l'alternative manuelle — joindre une photo existante, saisir l'adresse — et indiquez où se trouve le paramètre.

Le seul comportement qui garantit un refus définitif est de redemander en boucle. Enregistrez l'issue de chaque demande par capacité et par session, journalisez-la, et traitez un refus comme un état ordinaire de l'écran plutôt que comme une erreur.

Rapatrier les octets

Une photo prise avec un téléphone récent pèse de quatre à huit mégaoctets, et un technicien sur une connexion rurale en joint trois par intervention. Redimensionnez avant que les octets ne traversent le bridge : la requête au shell porte une limite sur le plus grand côté et une qualité JPEG, et le repli navigateur dessine le fichier choisi dans un canvas à la même limite et le réencode avant que **Upload** ne l'envoie. Six mégaoctets deviennent environ trois cents kilo-octets — la différence entre une application utilisable et inutilisable avec un signal faible.

Sur le serveur, travaillez en flux plutôt qu'en mémoire tampon : copiez le flux entrant vers le stockage et ne conservez que l'identifiant dans l'état de session, car un tableau d'octets stationné sur un objet d'écran est multiplié par chaque technicien connecté simultanément. Écrivez le fichier hors de la racine web sous un nom aléatoire, et servez-le via un gestionnaire qui vérifie que cette session a droit à cet ordre de travail.

La signature suit le même pipeline avec une source différente : une surface de dessin collecte les traits, l'image aplatie est exportée en PNG et envoyée, et le serveur la stocke avec l'identifiant de l'ordre de travail, le nom saisi par le signataire et un horodatage généré par le serveur — jamais une heure ou une identité fournie par le client. La localisation est la donnée la plus petite et la plus sensible : stockez ensemble la latitude, la longitude, le rayon de précision et l'heure du serveur, car une position à deux kilomètres près ne prouve pas l'arrivée, et ne la capturez que sur une action explicite.



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 :

- ✓ Appeler les fonctionnalités de l'appareil depuis le code côté serveur via le bridge du shell et recevoir les résultats sous forme d'événements.
- ✓ Vérifier les capacités à l'exécution pour que le même écran fonctionne dans un navigateur, une PWA et un shell natif.
- ✓ Demander les autorisations au bon moment et gérer un refus avec élégance.
- ✓ Transférer les photos, les signatures et les données de localisation vers le serveur de façon sûre et efficace.

Guide du lab / de l'examen

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



Lab pratique

Ajoutez des fonctionnalités de l'appareil à FieldOps : une action Take Photo qui utilise la caméra de l'appareil dans le shell natif et se replie sur un contrôle Upload dans le navigateur, une capture de signature envoyée sous forme d'image, une action Navigate qui lit la position de l'appareil et ouvre l'application de cartographie, une notification push lorsqu'un ordre de travail est attribué, et une vérification de capacités qui affiche ou masque chaque action selon le shell.

Point de départ

Vous ouvrez FieldOps tel que les Modules 1 à 4 l'ont laissé : la liste des ordres de travail, la page de détail et le flux de mise à jour du statut fonctionnent tous dans un simple navigateur, la PWA s'installe et se met à jour, le shell desktop WebView2 héberge l'application dans une fenêtre native, et le shell .NET MAUI l'exécute sur Android et iOS. La page de détail comporte déjà un panneau Photos, un panneau Signature et un bouton Navigate, tous des emplacements inertes, et les shells exposent un bridge que rien dans l'application web n'a encore appelé.

Approche suggérée

- ✓ Écrivez un service de capacités qui s'exécute une fois au démarrage de la session : demandez à la page hébergée si l'objet hôte du shell est présent, enregistrez dans la session les fonctionnalités qu'il annonce — caméra, localisation, notifications, biométrie — et exposez-les à chaque écran via une seule méthode.
- ✓ Implémentez Take Photo comme une seule action côté serveur reposant sur deux chemins : une requête au bridge limitée à 1600 pixels sur le plus grand côté avec une qualité JPEG de 70 dans les shells, et un contrôle **Upload** redimensionné dans le navigateur avant l'envoi ; stockez les octets renvoyés hors de la racine web et affichez la miniature via un gestionnaire autorisé.
- ✓ Ajoutez le bloc de signature sur une surface de dessin dans la page, capturez les traits, exportez le PNG aplati et envoyez-le avec l'identifiant de l'ordre de travail, le nom saisi par le signataire et un horodatage généré par le serveur.
- ✓ Reliez Navigate à une demande de localisation qui en affiche d'abord la raison, enregistrez la latitude, la longitude, le rayon de précision et l'heure du serveur dans l'enregistrement de la visite, puis passez la main à l'application de cartographie de la plateforme via le shell, ou ouvrez un lien de carte dans un nouvel onglet du navigateur lorsqu'il n'y a pas de bridge.
- ✓ Déclenchez une notification lorsqu'un ordre de travail est attribué — via le shell lorsque le système d'exploitation en fournit une, par web push sinon — demandez l'autorisation de notification après la première attribution de la session plutôt qu'à la connexion, et journalisez chaque issue d'autorisation par capacité.

Livrables attendus

- ✓ Capture photo avec repli sur l'envoi de fichier dans le navigateur

- ✓ Capture et envoi de la signature
- ✓ Géolocalisation et transfert vers l'application de cartographie
- ✓ Notification push à l'attribution
- ✓ Matrice de vérification des capacités par shell

Questions de révision

- ✓ Le résultat de la caméra arrive quarante secondes après le clic qui l'a demandé. Expliquez sur quelle requête il arrive, quel état de session il doit trouver pour savoir quoi faire des octets, et pourquoi l'image n'apparaît pas dans le navigateur sans push explicite.
- ✓ Un technicien a refusé définitivement l'autorisation de localisation il y a trois semaines sur son propre téléphone. Décrivez précisément ce que doit faire aujourd'hui un appui sur Navigate, et nommez la seule chose que l'application ne doit jamais faire en réponse.
- ✓ Take Photo fonctionne dans le shell mobile et le repli Upload fonctionne dans Chrome sur desktop. Nommez une chose réellement différente pour l'utilisateur entre ces deux chemins, et argumentez sur le caractère acceptable ou non de cette différence pour un technicien de terrain.



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.