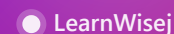


API del dispositivo tramite il bridge ibrido

Guida della lezione · Guida al lab / esame



Guida della lezione

Fotocamera, file, geolocalizzazione, notifiche push e biometria dall'app attraverso lo shell, con verifiche delle capacità e richieste di autorizzazione. Che cosa significa per un team che porta un'app Wisej.NET oltre la scheda del browser — e come affrontare questo modulo.



Focus di produzione

Fotocamera, file, geolocalizzazione, notifiche push e biometria dall'app attraverso lo shell, con verifiche delle capacità e richieste di autorizzazione.

1. Il problema in produzione

Un tecnico di FieldOps fotografa una valvola corrosa, raccoglie una firma sul vetro e guida fino all'indirizzo successivo — mentre il dispatcher che gli ha assegnato l'intervento osserva lo stesso ordine di lavoro in un browser desktop. Ogni controllo di FieldOps è un oggetto C# sul tuo server, che non può toccare la fotocamera, il chip GPS o l'area delle notifiche. Questo modulo è il giro di andata e ritorno che colma quel divario, e ciò che la schermata fa dove quel giro non esiste.

2. Concetti chiave

Concetto	Perché è importante
bridge del dispositivo	Il canale di messaggi che uno shell ibrido espone alla pagina che ospita: il tuo server chiede allo shell di eseguire un'operazione sul dispositivo, e lo shell risponde con un messaggio che l'app ritrasforma in un evento lato server.
host object	L'oggetto che una WebView inietta nella pagina ospitata così che il JavaScript lì presente possa chiamare codice nativo. Il C# lato server lo raggiunge solo tramite l'interop JavaScript dell'app.
descrittore delle capacità	Le funzionalità che uno shell dichiara all'avvio di una sessione — fotocamera, posizione, notifiche, biometria — messe in cache e consultate prima di mostrare qualsiasi azione sul dispositivo.
miglioramento progressivo	Progettare ogni schermata in modo che il percorso del semplice browser sia una funzionalità completa e il percorso nativo ne sia una versione migliore, mai l'unico che funziona.
autorizzazione just-in-time	Chiedere alla piattaforma l'accesso a fotocamera, posizione o notifiche nel momento in cui l'utente preme il pulsante che ne ha bisogno, subito dopo una frase che spiega perché.
evento di risultato del bridge	La risposta asincrona — byte della foto, coordinate o un rifiuto — che arriva su una richiesta successiva al clic che l'ha chiesta, e quindi ha bisogno di un proprio push.
archiviazione fuori dalla web root	Scrivere foto e firme acquisite dove nessun URL le espone, e poi servirle tramite un gestore che verifica se questa sessione può vedere quell'ordine di lavoro.

3. Il filo della lezione

Chiedere allo shell, e attendere la risposta

Il pulsante Take Photo non apre una fotocamera. Il suo gestore compone una richiesta — un'immagine fissa, lato più lungo 1600 pixel, qualità JPEG 70 — e la passa giù attraverso due salti. Il primo è quello che Wisej.NET ti dà già: il codice lato server esegue JavaScript nella pagina ospitata e ottiene un valore in risposta, il meccanismo dietro `Application.Call` e il livello di interop dei widget. Il secondo è quello dello shell: quel JavaScript chiama l'host object iniettato dalla WebView, e lo shell invoca la fotocamera della piattaforma. La risposta torna per la stessa strada, generata come un normale evento .NET che trasporta un array di byte.

Ciò che rende tutto questo diverso da un normale clic è il tempo. Il tecnico può impiegare quaranta secondi a inquadrare lo scatto, quindi il gestore non può bloccarsi; ritorna subito e la richiesta resta in sospeso in `Application.Session` sotto un token che hai generato. La risposta arriva su una richiesta diversa, quindi il codice che salva i byte e riempie la picture box gira fuori dall'evento originale, e le modifiche ai suoi controlli non raggiungeranno il browser da sole — concludi con `Application.Update`, esattamente come per il lavoro con `Application.StartTask`.

Anche annullamento ed errore sono risultati, non eccezioni: l'utente è tornato indietro, la richiesta è scaduta, lo shell è stato sospeso e ripreso. Arrivano tutti attraverso lo stesso evento, ed è il token che ti permette di scartare una risposta che nessuno sta più aspettando.

```
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);
}
```

Una schermata, quattro shell

FieldOps viene distribuita su un browser, una PWA, una finestra desktop WebView2 e uno shell mobile .NET MAUI, e la pagina dell'ordine di lavoro è la stessa pagina in tutti e quattro. Ciò che mostra deriva da una verifica delle capacità eseguita una volta per sessione: l'app chiede alla pagina se è presente un host object e che cosa lo shell dichiara di saper fare, e quella risposta finisce nella sessione perché ogni schermata la legga. Ciò che non devi fare è analizzare lo user agent — descrive il browser, non se questa build del tuo shell espone una fotocamera.

La matrice ha una riga per capacità e una colonna per shell. La fotocamera è nativa nello shell mobile e, nel browser di un telefono, è il controllo **Upload**, che il browser soddisfa direttamente dall'app della fotocamera; lo shell desktop offre invece una finestra di dialogo dei file. La posizione è nativa su mobile e altrove è la Geolocation API del browser raggiunta tramite interop JavaScript. Le notifiche arrivano dal sistema operativo nei due shell e dal web push nel browser e nella PWA. La biometria esiste solo dove la piattaforma ha un autenticatore, quindi FieldOps la usa per sbloccare di nuovo una sessione, mai come unico modo di accedere.

Take Photo e **Upload** sono quindi due rappresentazioni di un'unica azione, e il resto della schermata non sa mai quale sia stata eseguita. Una capacità che lo shell non dichiara semplicemente non compare: un pulsante disabilitato senza spiegazione è peggio di uno assente, e uno che genera un'eccezione quando lo premi è peggio di entrambi.

L'autorizzazione nel momento dell'uso

Davanti a ogni funzionalità del dispositivo ci sono due cancelli: l'autorizzazione della piattaforma, con cui il sistema operativo o il browser chiede all'utente di consentire fotocamera, posizione o notifiche, e la tua spiegazione, che viene prima. FieldOps mostra una frase — la posizione mostra la distanza dal prossimo sito e registra l'orario di arrivo — e solo dopo attiva la richiesta della piattaforma.

Il momento decide la risposta. Una richiesta di posizione sulla schermata di login, prima che il tecnico abbia visto un ordine di lavoro, riceve un "no", e su diverse piattaforme quel no è permanente; la stessa

richiesta generata da Navigate su un intervento il cui indirizzo è sullo schermo riceve un "sì" quasi sempre. Per questo FieldOps chiede le notifiche dopo la prima assegnazione della sessione, e la fotocamera quando viene premuto Take Photo.

- Concessa: prosegui, e ricorda la concessione per questa sessione.
- Negata una volta: mantieni l'azione e il motivo; il prossimo intervento potrebbe ottenere una risposta diversa.
- Negata in modo permanente: sostituisci l'alternativa manuale — allega una foto esistente, digita l'indirizzo — e mostra dove si trova l'impostazione.

L'unico comportamento che garantisce un rifiuto permanente è ripetere la richiesta in un ciclo. Registra l'esito di ogni richiesta per capacità e per sessione, scrivilo nel log, e tratta un rifiuto come uno stato ordinario della schermata invece che come un errore.

Portare i byte a casa

Una foto da un telefono attuale pesa da quattro a otto megabyte, e un tecnico su una connessione di campagna ne allega tre per intervento. Ridimensiona prima che i byte attraversino il bridge: la richiesta allo shell porta un limite al lato più lungo e una qualità JPEG, e il ripiego per il browser disegna il file scelto in un canvas con lo stesso limite e lo ricodifica prima che **Upload** lo invii. Sei megabyte diventano circa trecento kilobyte — la differenza tra un'app utilizzabile e una inutilizzabile con un segnale debole.

Sul server, usa lo streaming invece del buffering: copia lo stream in ingresso nell'archivio e tieni nello stato di sessione solo l'identificativo, perché un array di byte parcheggiato su un oggetto della schermata viene moltiplicato per ogni tecnico connesso contemporaneamente. Scrivi il file fuori dalla web root con un nome casuale, e servilo tramite un gestore che verifica che questa sessione abbia diritto a quell'ordine di lavoro.

La firma segue la stessa pipeline con una sorgente diversa: una superficie di disegno raccoglie i tratti, l'immagine appiattita viene esportata come PNG e inviata, e il server la salva con l'identificativo dell'ordine di lavoro, il nome digitato del firmatario e un timestamp generato dal server — mai un orario o un'identità forniti dal client. La posizione è il payload più piccolo e più sensibile: salva insieme latitudine, longitudine, raggio di accuratezza e ora del server, perché una posizione con due chilometri di approssimazione non prova l'arrivo, e acquisiscila solo su un'azione esplicita.



La regola del packaging

L'app resta un'app web Wisej.NET servita dal tuo server; lo shell è un veicolo di distribuzione. Mantieni ogni funzionalità funzionante in un semplice browser, poi aggiungi le capacità del dispositivo come miglioramenti progressivi dietro una verifica delle capacità.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Chiamare le funzionalità del dispositivo dal codice lato server tramite il bridge dello shell e ricevere i risultati come eventi.

- ✓ Verificare le capacità a runtime così che la stessa schermata funzioni in un browser, in una PWA e in uno shell nativo.
- ✓ Richiedere le autorizzazioni al momento giusto e gestire il rifiuto con eleganza.
- ✓ Trasferire al server foto, firme e dati di posizione in modo sicuro ed efficiente.

Guida al lab / esame

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



Lab pratico

Aggiungi le funzionalità del dispositivo a FieldOps: un'azione Take Photo che usa la fotocamera del dispositivo nello shell nativo e ripiega su un controllo Upload nel browser, un'acquisizione della firma caricata come immagine, un'azione Navigate che legge la posizione del dispositivo e apre l'app delle mappe, una notifica push quando viene assegnato un ordine di lavoro, e una verifica delle capacità che mostra o nasconde ogni azione a seconda dello shell.

Punto di partenza

Apri FieldOps come l'hanno lasciata i Moduli dall'1 al 4: l'elenco degli ordini di lavoro, la pagina di dettaglio e il flusso di aggiornamento dello stato girano tutti in un semplice browser, la PWA si installa e si aggiorna da sola, lo shell desktop WebView2 ospita l'app in una finestra nativa, e lo shell .NET MAUI la esegue su Android e iOS. La pagina di dettaglio ha già un pannello Photos, un pannello Signature e un pulsante Navigate, tutti segnaposto inerti, e gli shell espongono un bridge che nulla nell'app web ha ancora chiamato.

Approccio suggerito

- ✓ Scrivi un servizio delle capacità che viene eseguito una volta all'avvio della sessione: chiedi alla pagina ospitata se l'host object dello shell è presente, registra nella sessione quali funzionalità dichiara — fotocamera, posizione, notifiche, biometria — ed esponile a ogni schermata tramite un unico metodo.
- ✓ Implementa Take Photo come un'unica azione lato server con due percorsi dietro: negli shell una richiesta al bridge limitata a 1600 pixel sul lato più lungo e qualità JPEG 70, e nel browser un controllo **Upload** ridimensionato prima dell'invio; salva i byte restituiti fuori dalla web root e mostra la miniatura tramite un gestore autorizzato.
- ✓ Aggiungi il riquadro per la firma su una superficie di disegno nella pagina, acquisisci i tratti, esporta il PNG appiattito e caricalo con l'identificativo dell'ordine di lavoro, il nome digitato del firmatario e un timestamp generato dal server.
- ✓ Collega Navigate a una richiesta di posizione che mostra prima il motivo, salva sul record della visita latitudine, longitudine, raggio di accuratezza e ora del server, poi passa all'app delle mappe della piattaforma tramite lo shell, oppure apri un link alle mappe in una nuova scheda del browser quando non c'è un bridge.
- ✓ Genera una notifica quando viene assegnato un ordine di lavoro — tramite lo shell dove il sistema operativo ne offre una, altrimenti con il web push — chiedi l'autorizzazione alle notifiche dopo la prima assegnazione della sessione invece che al login, e registra nel log ogni esito delle autorizzazioni per capacità.

Deliverable richiesti

- ✓ Acquisizione dalla fotocamera con ripiego sull'upload nel browser

- ✓ Acquisizione e caricamento della firma
- ✓ Geolocalizzazione e passaggio all'app delle mappe
- ✓ Notifica push all'assegnazione
- ✓ Matrice delle verifiche delle capacità tra gli shell

Domande di revisione per lo studente

- ✓ Il risultato della fotocamera arriva quaranta secondi dopo il clic che l'ha chiesto. Spiega su quale richiesta arriva, quale stato di sessione deve trovare per sapere che cosa fare dei byte, e perché la picture box non compare nel browser senza un push esplicito.
- ✓ Tre settimane fa un tecnico ha negato in modo permanente l'autorizzazione alla posizione sul proprio telefono. Descrivi esattamente che cosa dovrebbe fare oggi la pressione di Navigate, e indica l'unica cosa che l'app non deve mai fare in risposta.
- ✓ Take Photo funziona nello shell mobile e il ripiego su Upload funziona in Chrome desktop. Indica una cosa davvero diversa per l'utente tra questi due percorsi, e argomenta se quella differenza sia accettabile per un tecnico sul campo.



La regola del packaging

L'app resta un'app web Wisej.NET servita dal tuo server; lo shell è un veicolo di distribuzione. Mantieni ogni funzionalità funzionante in un semplice browser, poi aggiungi le capacità del dispositivo come miglioramenti progressivi dietro una verifica delle capacità.