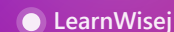


Shell mobile con .NET MAUI

Guida della lezione · Guida al lab / esame



Guida della lezione

Ospitare l'app in uno shell ibrido .NET MAUI per Android e iOS, navigazione e comportamento del pulsante Indietro, ciclo di vita dell'app e peculiarità delle piattaforme. 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

Ospitare l'app in uno shell ibrido .NET MAUI per Android e iOS, navigazione e comportamento del pulsante Indietro, ciclo di vita dell'app e peculiarità delle piattaforme.

1. Il problema in produzione

I tecnici di FieldOps lavorano da un telefono in una sala macchine, in uno scantinato o in un furgone, e una scheda del browser è un contenitore scadente per questo: nessuna icona nella schermata home da distribuire, nessuno store o canale enterprise, e nessuna strada verso la fotocamera e il GPS di cui avrà bisogno il Modulo 5. Il sistema operativo mobile è anche un ospite molto più aggressivo — sospende l'app nell'istante in cui arriva una chiamata, la ridisegna sotto un notch e consegna al tecnico un gesto Indietro di sistema che FieldOps non ha mai chiesto.

2. Concetti chiave

Concetto	Perché è importante
shell ibrido .NET MAUI	Una piccola app nativa per Android e iOS la cui unica pagina ospita la tua applicazione Wisej.NET in una web view, così che il tecnico veda proprio le schermate C# che già rilasci.
host web view nativo	Il motore browser della piattaforma incorporato in quella pagina — Android System WebView, WKWebView su iOS — che esegue lo stesso client che riceve un browser desktop, ed è per questo che nessuna schermata va riscritta per il mobile.
bridge ibrido	Il canale bidirezionale tra la pagina ospitata e il codice nativo, così che il C# lato server possa chiedere qualcosa al dispositivo e ricevere la risposta come evento invece che come navigazione di pagina.
ciclo di vita dell'applicazione MAUI	Gli eventi che il sistema operativo genera attorno all'app — finestra creata, attivata, disattivata, fermata e ripresa, più <code>OnSleep</code> e <code>OnResume</code> — nessuno dei quali viene generato da una scheda del browser.
finestra di sopravvivenza della sessione	L'intervallo di tempo in background durante il quale la sessione Wisej.NET è ancora viva sul server, così che una ripresa possa riconnettersi allo stato esistente invece di far ripartire l'app da zero.
inset della safe area	Il padding specifico di ogni dispositivo che tiene i contenuti lontani dal notch, dalla barra di stato e dalla striscia dell'indicatore home, comunicato dalla piattaforma invece che dedotto dalla dimensione dello schermo.
mappatura della navigazione Indietro	La regola che decide che cosa fa il gesto Indietro di Android: tornare indietro nella navigazione interna dell'app, oppure chiedere conferma prima di uscire — mai distruggere in silenzio un ordine di lavoro compilato a metà.

3. Il filo della lezione

Un progetto di shell, due store, lo stesso server

Lo shell mobile di FieldOps è un'applicazione .NET MAUI con un solo compito: creare una finestra, ospitare una web view, puntarla al server di FieldOps. Il supporto Wisej.NET Hybrid fornisce quello shell come progetto da compilare e adattare invece che da scrivere da zero, e la pagina che ospita è la normale applicazione FieldOps del Modulo 1. Nulla cambia in `Wisej.Web`, nel designer di Visual Studio o in `Application.Session`; i controlli vivono ancora sul server e la web view ne rende ancora l'immagine speculare.

Un unico progetto MAUI punta ad Android e iOS contemporaneamente, e le cartelle di piattaforma al suo interno contengono solo ciò che è davvero diverso: l'activity e il manifest di Android, l'`Info.plist` e gli entitlement di iOS, le icone e gli splash screen. Tieni l'indirizzo del server in un unico valore di configurazione che lo shell legge all'avvio, così il debug punta a un portatile, il test allo staging e la release alla produzione senza modificare il codice.

Quella separazione è anche l'economia dello shell. FieldOps rilascia nuovo codice server nel momento in cui fai il deploy; lo shell va ricompilato solo quando cambia lo shell stesso — una nuova icona, una nuova autorizzazione, una nuova capacità del bridge. Tutto ciò che può vivere sul server dovrebbe starci, perché il server è la parte che puoi correggere oggi pomeriggio mentre la coda di revisione di uno store si misura in giorni.

Il passaggio in background è il caso normale, non il caso di errore

Su un telefono l'app viene interrotta di continuo: una chiamata, un codice ricambio verificato in un'altra app, uno schermo bloccato mentre si sale su una scala. Ognuna di queste genera la sequenza del ciclo di vita MAUI — la finestra disattivata, poi fermata, poi ripresa al ritorno, con gli override `OnSleep` e `OnResume` a livello di applicazione che scattano insieme. Gestiscili in modo esplicito, perché il comportamento predefinito di una web view rimossa dalla memoria è ricaricare la pagina, e un ricaricamento è una nuova sessione Wisej.NET: la lettura del contatore digitata a metà è sparita.

Il progetto che funziona tratta un breve background come una pausa e uno lungo come un riavvio. Alla fermata, registra l'orario e lascia la sessione inattiva. Alla ripresa, confronta il tempo trascorso con il timeout di sessione configurato per il server di FieldOps; se la sessione dovrebbe essere ancora viva, riconnettiti e lascia che il server invii di nuovo lo stato corrente invece di far navigare di nuovo la web view. Se l'intervallo supera il timeout, o la piattaforma segnala che l'app è stata terminata per mancanza di memoria, ripristina in modo deliberato: ricarica, poi riapri l'ordine di lavoro usando un identificativo che lo shell ha salvato in locale. Tratta un avviso di memoria insufficiente esattamente come una fermata.

```
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(); // riconnettiti, non ricaricare
    return window;
}

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

Il lato server del contratto conta altrettanto: mantieni il timeout di sessione di FieldOps abbastanza lungo da coprire un'interruzione realistica, e fai in modo che l'editor degli ordini di lavoro salvi una bozza a ogni modifica di campo invece che solo all'invio.

Indietro, notch e la striscia in basso

Il gesto Indietro di Android è un evento di sistema, e ignorarlo significa accettare il comportamento predefinito della piattaforma: l'activity si chiude, quindi un tecnico che fa swipe indietro da una pagina di dettaglio dell'ordine di lavoro perde l'app invece di tornare all'elenco. Intercetta la navigazione Indietro nello shell e instradala prima all'app web — la pagina di dettaglio torna all'elenco, una finestra di dialogo aperta si chiude — passando a una richiesta di conferma solo quando non resta nulla da chiudere. Lo stato che risponde alla domanda "c'è qualcosa a cui tornare" vive sul server, quindi lo shell deve chiederlo all'app invece di indovinare: tieni uno stack di navigazione piccolo ed esplicito dentro FieldOps stessa. iOS non ha un pulsante Indietro, quindi il comando Indietro proprio dell'app deve esistere in ogni caso.

Le safe area sono l'altra peculiarità visibile delle piattaforme. Un telefono con notch riserva la striscia superiore e un iPhone moderno riserva in basso una striscia per l'indicatore home; i contenuti disegnati lì sono nascosti o impossibili da toccare. Leggi gli inset comunicati dalla piattaforma e applicali come padding attorno ai contenuti ospitati, così che la barra dell'app di FieldOps stia sotto la barra di stato e i pulsanti di aggiornamento dello stato stiano sopra l'indicatore home. Applica il padding all'host invece di scrivere valori fissi in pixel nel layout Wisej.NET, e le stesse schermate restano corrette in un semplice browser, su un dispositivo senza notch e in orizzontale.

Emulatori, dispositivi e due comandi di build

Il ciclo di build è specifico per piattaforma e conviene prepararlo prima di scrivere qualsiasi codice dello shell. Android richiede l'SDK e almeno un'immagine dell'Android Emulator; un'esecuzione di debug è `dotnet build -t:Run -f net8.0-android` o il target equivalente in Visual Studio. iOS richiede un Mac nel ciclo — l'iOS Simulator e una build come `dotnet build -f net8.0-ios` — anche quando sviluppi su Windows e ti colleghi a quel Mac da remoto. L'hot restart accorcia il ciclo, e poiché la maggior parte delle modifiche a FieldOps è lato server, molte non richiedono alcuna ricompilazione: cambia il C#, riavvia il server, trascina verso il basso nello shell.

Il debug attraversa due runtime: i breakpoint nel C# dello shell coprono ciclo di vita e chiamate al bridge, i breakpoint nel codice server coprono le schermate. Quando l'emulatore non riesce a raggiungere il tuo server di sviluppo, la causa è quasi sempre l'indirizzo di loopback — il `localhost` di un emulatore è il dispositivo emulato, non la tua macchina — oppure un certificato autofirmato che la piattaforma rifiuta. Poi metti alla prova lo shell su hardware reale: aree di tocco, tastiera che copre un campo e rimozione dalla memoria sotto pressione si comportano tutti in modo diverso lì, e gli screenshot del lab da entrambi gli emulatori sono la base con cui confronti quell'esecuzione sul dispositivo.



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:

- ✓ Ospitare l'app Wisej.NET in uno shell ibrido .NET MAUI per Android e iOS.
- ✓ Gestire gli eventi del ciclo di vita mobile — background, ripresa, memoria insufficiente — senza perdere la sessione.
- ✓ Mappare sull'app web la navigazione della piattaforma, il pulsante Indietro e le safe area.
- ✓ Configurare emulatori, debug sui dispositivi e la pipeline di build per entrambe le piattaforme.

Guida al lab / esame

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



Lab pratico

Costruisci lo shell mobile di FieldOps con .NET MAUI: una pagina ibrida che ospita l'app su Android e iOS, una sessione che sopravvive al passaggio in background e riprende al ritorno, il pulsante Indietro di Android mappato sulla navigazione interna dell'app, un padding di safe area per i dispositivi con notch, e una build di debug in esecuzione su un emulatore per piattaforma con gli screenshot dell'elenco degli ordini di lavoro.

Punto di partenza

Apri la soluzione FieldOps del Modulo 1 — l'elenco degli ordini di lavoro, la pagina di dettaglio dell'ordine e il flusso di aggiornamento dello stato, già in esecuzione in un semplice browser e, dal Modulo 2, installabili come PWA. Nulla in quelle schermate cambia in questo lab; accanto al progetto web aggiungi un secondo head, uno shell .NET MAUI che ospita sul telefono la stessa app resa dal server.

Approccio suggerito

- ✓ Aggiungi il progetto di shell MAUI alla soluzione FieldOps, imposta i target Android e iOS, e leggi l'indirizzo del server dalla configurazione così che le build di debug e di release puntino a host diversi.
- ✓ Ospita l'app FieldOps nella web view dello shell e verifica che l'elenco degli ordini di lavoro, la pagina di dettaglio e un aggiornamento dello stato funzionino tutti dall'inizio alla fine prima di toccare qualsiasi cosa specifica della piattaforma.
- ✓ Gestisci gli eventi di finestra fermata, ripresa e memoria insufficiente: registra l'orario in cui l'app è passata in background e, a una ripresa entro il timeout di sessione, riconnettiti invece di ricaricare la pagina.
- ✓ Intercetta la navigazione Indietro di Android e instradala nello stack di navigazione di FieldOps, chiedendo conferma solo quando non resta nulla a cui tornare; applica gli inset della safe area comunicati come padding attorno ai contenuti ospitati.
- ✓ Esegui una build di debug sull'Android Emulator e sull'iOS Simulator, cattura l'elenco degli ordini di lavoro su ciascuno, e annota ogni differenza di cui hai dovuto tenere conto per piattaforma.

Deliverable richiesti

- ✓ Progetto di shell ibrido .NET MAUI per Android e iOS
- ✓ Gestione del ciclo di vita per background e ripresa
- ✓ Mappatura del pulsante Indietro e della safe area
- ✓ Build sugli emulatori per entrambe le piattaforme
- ✓ Screenshot da entrambi gli emulatori

Domande di revisione per lo studente

- ✓ Un tecnico è a metà di un aggiornamento dello stato quando arriva una chiamata di tre minuti. Che cosa deve essere vero sul server e nello shell perché l'aggiornamento lasciato a metà sia ancora lì dopo, e che cosa cambia se la chiamata dura un'ora?
- ✓ Lo shell non sa nulla di quale schermata di FieldOps sia aperta. Come fa un gesto Indietro a scegliere tra chiudere la pagina di dettaglio, chiudere una finestra di dialogo e uscire dall'app, e dove vive davvero quello stato?
- ✓ Quali modifiche di questo modulo noterebbe ancora un tecnico se aprisse FieldOps nel browser del telefono invece che nello shell, e quali sparirebbero? Che cosa ti dice questo su dove mettere il padding della safe area?



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.