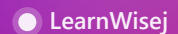


Shells móviles con .NET MAUI

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Alojar la aplicación en un shell híbrido de .NET MAUI para Android e iOS, la navegación y el comportamiento del botón Atrás, el ciclo de vida de la aplicación y las particularidades de cada plataforma. Qué significa esto para un equipo que lleva una aplicación Wisej.NET más allá de la pestaña del navegador, y cómo leer este módulo.



Enfoque de producción

Alojar la aplicación en un shell híbrido de .NET MAUI para Android e iOS, la navegación y el comportamiento del botón Atrás, el ciclo de vida de la aplicación y las particularidades de cada plataforma.

1. El problema en producción

Los técnicos de FieldOps trabajan con un teléfono en una sala de máquinas, un sótano o una furgoneta, y una pestaña del navegador es un mal contenedor para eso: no hay icono en la pantalla de inicio que repartir, ni canal de tienda o empresarial, ni acceso a la cámara y al GPS que necesitará el Módulo 5. Además, el sistema operativo móvil es un anfitrión mucho más agresivo: suspende la aplicación en cuanto entra una llamada, la vuelve a dibujar bajo un notch y le da al técnico un gesto Atrás del sistema que FieldOps nunca pidió.

2. Conceptos clave

Concepto	Por qué importa
shell híbrido de .NET MAUI	Una pequeña aplicación nativa para Android e iOS cuya única página aloja tu aplicación Wisej.NET en una vista web, de modo que lo que ve el técnico son las pantallas C# que ya publicas.
vista web nativa anfitriona	El motor de navegador de la plataforma integrado en esa página (Android System WebView, WKWebView en iOS), que ejecuta el mismo cliente que recibe un navegador de escritorio; por eso ninguna pantalla se reescribe para móvil.
punto híbrido	El canal bidireccional entre la página alojada y el código nativo, para que el código C# del servidor pueda pedir algo al dispositivo y recibir la respuesta como un evento en lugar de como una navegación de página.
ciclo de vida de la aplicación MAUI	Los eventos que el sistema operativo genera alrededor de la aplicación (ventana creada, activada, desactivada, detenida y reanudada, más <code>OnSleep</code> y <code>OnResume</code>), ninguno de los cuales genera una pestaña del navegador.
margen de supervivencia de la sesión	El tiempo en segundo plano durante el cual la sesión de Wisej.NET sigue viva en el servidor, de modo que una reanudación puede reconectar con el estado existente en lugar de volver a iniciar la aplicación.
márgenes del área segura	El relleno específico de cada dispositivo que mantiene el contenido fuera del notch, de la barra de estado y de la franja del indicador de inicio, indicado por la plataforma y no adivinado a partir del tamaño de la pantalla.
asignación de la navegación Atrás	La regla que decide qué hace el gesto Atrás de Android: retroceder dentro de la navegación propia de la aplicación o confirmar la salida, pero nunca destruir en silencio una orden de trabajo a medio rellenar.

3. Desarrollo de la lección

Un proyecto de shell, dos tiendas, el mismo servidor

El shell móvil de FieldOps es una aplicación .NET MAUI con una sola tarea: crear una ventana, alojar una vista web y apuntarla al servidor de FieldOps. El soporte de Wisej.NET Hybrid proporciona ese shell como un proyecto que compilas y ajustas en lugar de escribirlo desde cero, y la página que aloja es la aplicación FieldOps normal del Módulo 1. Nada cambia en `Wisej.Web`, en el diseñador de Visual Studio ni en `Application.Session`; los controles siguen viviendo en el servidor y la vista web sigue renderizando su reflejo.

Un único proyecto MAUI se dirige a la vez a Android y a iOS, y las carpetas de plataforma que contiene solo guardan lo que realmente difiere: la actividad y el manifiesto de Android, el `Info.plist` y los entitlements de iOS, los iconos y las pantallas de bienvenida. Guarda la dirección del servidor en un único valor de configuración que el shell lee al iniciar, para que la depuración apunte a un portátil, las pruebas a staging y la publicación a producción sin cambiar el código.

Esa separación es también la economía del shell. FieldOps publica código nuevo del servidor en cuanto lo despliegas; el shell solo necesita recompilarse cuando cambia el propio shell: un icono nuevo, un permiso nuevo, una capacidad nueva del puente. Todo lo que pueda estar en el servidor debería estar ahí, porque el servidor es la parte que puedes corregir esta misma tarde, mientras que la cola de revisión de una tienda se mide en días.

El segundo plano es el caso normal, no el caso de error

En un teléfono la aplicación se interrumpe constantemente: una llamada, un número de pieza consultado en otra aplicación, una pantalla bloqueada mientras se sube una escalera. Cada interrupción genera la secuencia del ciclo de vida de MAUI (la ventana se desactiva, después se detiene y al volver se reanuda), mientras se disparan a la vez las sobrescrituras `OnSleep` y `OnResume` a nivel de aplicación. Géstionalas de forma explícita, porque el comportamiento predeterminado de una vista web expulsada de la memoria es recargar la página, y una recarga es una sesión nueva de Wisej.NET: la lectura del contador a medio escribir desaparece.

El diseño que funciona trata un paso breve a segundo plano como una pausa y uno largo como un reinicio. Al detenerse, registra la hora y deja la sesión inactiva. Al reanudar, compara el tiempo transcurrido con el tiempo de espera de sesión configurado en el servidor de FieldOps; si la sesión debería seguir viva, reconecta y deja que el servidor devuelva el estado actual en lugar de volver a navegar en la vista web. Si el intervalo supera el tiempo de espera, o la plataforma indica que la aplicación se terminó por falta de memoria, restaura de forma deliberada: recarga y vuelve a abrir la orden de trabajo usando un identificador que el shell guardó en local. Trata un aviso de memoria baja exactamente igual que una detención.

```
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();
```

La parte del servidor del contrato importa igual: mantén el tiempo de espera de la sesión de FieldOps lo bastante largo para cubrir una interrupción realista, y haz que el editor de órdenes de trabajo guarde un borrador en cada cambio de campo y no solo al enviar.

Atrás, notches y la franja inferior

El gesto Atrás de Android es un evento del sistema, e ignorarlo significa aplicar el comportamiento predeterminado de la plataforma: la actividad se cierra, así que un técnico que desliza hacia atrás desde la página de detalle de una orden de trabajo sale de la aplicación en lugar de volver a la lista. Intercepta la navegación Atrás en el shell y llévala primero a la aplicación web: la página de detalle vuelve a la lista, un cuadro de diálogo abierto se cierra, y solo se recurre a una solicitud de confirmación cuando ya no queda nada que retirar. El estado que responde a «¿hay algo a lo que volver?» vive en el servidor, así que el shell

debe preguntar a la aplicación en lugar de adivinar: mantén en el propio FieldOps una pila de navegación pequeña y explícita. iOS no tiene botón Atrás, así que el control propio de la aplicación para volver debe existir en cualquier caso.

Las áreas seguras son la otra particularidad visible de la plataforma. Un teléfono con notch reserva la franja superior y un iPhone moderno reserva abajo una franja para el indicador de inicio; el contenido dibujado ahí queda oculto o no se puede tocar. Lee los márgenes que indica la plataforma y aplícalos como relleno alrededor del contenido alojado, para que la barra de la aplicación de FieldOps quede debajo de la barra de estado y los botones de actualización del estado queden encima del indicador de inicio. Aplica el relleno al anfitrión en lugar de escribir valores fijos en píxeles en el diseño de Wisej.NET, y las mismas pantallas seguirán siendo correctas en un navegador normal, en un dispositivo sin notch y en horizontal.

Emuladores, dispositivos y dos comandos de compilación

El ciclo de compilación es distinto en cada plataforma y conviene prepararlo antes de escribir código del shell. Android necesita el SDK y al menos una imagen del Android Emulator; una ejecución de depuración es `dotnet build -t:Run -f net8.0-android` o el destino equivalente en Visual Studio. iOS necesita un Mac en el proceso (el iOS Simulator y una compilación como `dotnet build -f net8.0-ios`) aunque desarrolles en Windows y te emparejes con ese Mac de forma remota. Hot restart acorta el ciclo, y como la mayoría de los cambios de FieldOps son del lado del servidor, muchos no requieren ninguna recompilación: cambia el C#, reinicia el servidor y desliza hacia abajo en el shell para recargar.

La depuración abarca dos runtimes: los puntos de interrupción en el C# del shell cubren el ciclo de vida y las llamadas al puente, y los del código del servidor cubren las pantallas. Cuando el emulador no llega a tu servidor de desarrollo, la causa casi siempre es la dirección de loopback (el `localhost` de un emulador es el dispositivo emulado, no tu máquina) o un certificado autofirmado que la plataforma rechaza. Después demuestra el shell en hardware real: los objetivos táctiles, el teclado que tapa un campo y la expulsión por presión de memoria se comportan de otra forma ahí, y las capturas de ambos emuladores del laboratorio son la referencia con la que comparas esa ejecución en el dispositivo.



Regla de empaquetado

La aplicación sigue siendo una aplicación web Wisej.NET servida por tu servidor; el shell es un medio de distribución. Mantén todas las funciones operativas en un navegador normal y añade después las capacidades del dispositivo como mejoras progresivas detrás de una comprobación de capacidades.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Alojar la aplicación Wisej.NET en un shell híbrido de .NET MAUI para Android e iOS.
- ✓ Gestionar los eventos del ciclo de vida móvil (segundo plano, reanudación, memoria baja) sin perder la sesión.
- ✓ Asignar la navegación de la plataforma, el botón Atrás y las áreas seguras a la aplicación web.
- ✓ Configurar los emuladores, la depuración en dispositivos y el proceso de compilación para ambas plataformas.

Guía del laboratorio / examen

Qué construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Construye el shell móvil de FieldOps con .NET MAUI: una página híbrida que aloje la aplicación en Android e iOS, una sesión que sobreviva al paso a segundo plano y se reanude al volver, el botón Atrás de Android asignado a la navegación dentro de la aplicación, márgenes de área segura para dispositivos con notch, y una compilación de depuración ejecutándose en un emulador por plataforma con capturas de pantalla de la lista de órdenes de trabajo.

Punto de partida

Abres la solución FieldOps del Módulo 1: la lista de órdenes de trabajo, la página de detalle de la orden de trabajo y el flujo de actualización del estado, que ya funcionan en un navegador normal y, desde el Módulo 2, se pueden instalar como PWA. Nada de esas pantallas cambia en este laboratorio; añades un segundo proyecto de destino junto al proyecto web, un shell .NET MAUI que aloja en un teléfono la misma aplicación renderizada por el servidor.

Enfoque sugerido

- ✓ Añade el proyecto de shell MAUI a la solución FieldOps, dirígelo a Android e iOS y lee la dirección del servidor de la configuración para que las compilaciones de depuración y de publicación apunten a hosts distintos.
- ✓ Aloja la aplicación FieldOps en la vista web del shell y confirma que la lista de órdenes de trabajo, la página de detalle y una actualización de estado funcionan de principio a fin antes de tocar nada específico de una plataforma.
- ✓ Gestiona los eventos de detención, reanudación y memoria baja de la ventana: registra la hora en que la aplicación pasó a segundo plano y, al reanudar dentro del tiempo de espera de la sesión, reconecta en lugar de recargar la página.
- ✓ Intercepta la navegación Atrás de Android y llévala a la pila de navegación de FieldOps, pidiendo confirmación solo cuando no quede nada a lo que volver; aplica los márgenes del área segura indicados como relleno alrededor del contenido alojado.
- ✓ Ejecuta una compilación de depuración en el Android Emulator y en el iOS Simulator, captura la lista de órdenes de trabajo en cada uno y anota todas las diferencias que has tenido que tener en cuenta en cada plataforma.

Entregables requeridos

- ✓ Proyecto de shell híbrido de .NET MAUI para Android e iOS
- ✓ Gestión del ciclo de vida para el segundo plano y la reanudación
- ✓ Asignación del botón Atrás y del área segura
- ✓ Compilaciones en emulador para ambas plataformas
- ✓ Capturas de pantalla de ambos emuladores

Preguntas de repaso

- ✓ Un técnico está a mitad de una actualización de estado cuando recibe una llamada de tres minutos. ¿Qué debe cumplirse en el servidor y en el shell para que la actualización a medias siga ahí después, y qué cambia si la llamada dura una hora?
- ✓ El shell no sabe qué pantalla de FieldOps está abierta. ¿Cómo decide un gesto Atrás entre volver desde la página de detalle, cerrar un cuadro de diálogo y salir de la aplicación, y dónde está realmente ese estado?
- ✓ ¿Cuáles de los cambios de este módulo seguiría notando un técnico si abriera FieldOps en el navegador del teléfono en lugar de en el shell, y cuáles desaparecerían? ¿Qué te dice eso sobre dónde poner el relleno del área segura?



Regla de empaquetado

La aplicación sigue siendo una aplicación web Wisej.NET servida por tu servidor; el shell es un medio de distribución. Mantén todas las funciones operativas en un navegador normal y añade después las capacidades del dispositivo como mejoras progresivas detrás de una comprobación de capacidades.