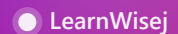


API del dispositivo a través del puente híbrido

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Cámara, archivos, geolocalización, notificaciones push y biometría desde la aplicación a través del shell, con comprobaciones de capacidades y solicitudes de permisos. 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

Cámara, archivos, geolocalización, notificaciones push y biometría desde la aplicación a través del shell, con comprobaciones de capacidades y solicitudes de permisos.

1. El problema en producción

Un técnico de FieldOps fotografía una válvula corroída, recoge una firma sobre la pantalla y conduce hasta la siguiente dirección, mientras el operador que le asignó el trabajo sigue la misma orden de trabajo en un navegador de escritorio. Cada control de FieldOps es un objeto C# en tu servidor, que no puede tocar la cámara, el chip GPS ni la bandeja de notificaciones. Este módulo es el viaje de ida y vuelta que cierra esa brecha, y lo que hace la pantalla donde ese viaje no existe.

2. Conceptos clave

Concepto	Por qué importa
puente con el dispositivo	El canal de mensajes que un shell híbrido expone a la página que aloja: tu servidor pide al shell que realice una operación del dispositivo, y el shell responde con un mensaje que la aplicación convierte de nuevo en un evento del servidor.
objeto host	El objeto que un WebView inyecta en la página alojada para que el JavaScript de esa página pueda llamar a código nativo. El código C# del servidor solo llega a él mediante la interoperabilidad con JavaScript de la aplicación.
descriptor de capacidades	Las funciones que comunica un shell al iniciar una sesión (cámara, ubicación, notificaciones, biometría), guardadas y consultadas antes de mostrar cualquier acción del dispositivo.
mejora progresiva	Diseñar cada pantalla para que la vía del navegador normal sea una función completa y la vía nativa sea una versión mejor de ella, nunca la única que funciona.
permiso en el momento justo	Pedir a la plataforma acceso a la cámara, la ubicación o las notificaciones en el momento en que el usuario pulsa el botón que lo necesita, justo después de una frase que explica por qué.
evento de resultado del puente	La respuesta asíncrona (bytes de la foto, coordenadas o una negativa) que llega en una solicitud posterior al clic que la pidió, por lo que necesita su propio envío al navegador.
almacenamiento fuera de la raíz web	Escribir las fotos y firmas capturadas donde ninguna URL apunta a ellas, y servir las después mediante un controlador que comprueba si esta sesión puede ver esa orden de trabajo.

3. Desarrollo de la lección

Pedírselo al shell y esperar la respuesta

El botón Take Photo no abre una cámara. Su controlador compone una solicitud (una imagen fija, 1600 píxeles en el lado más largo, calidad JPEG 70) y la envía a través de dos saltos. El primero es el que Wisej.NET ya te ofrece: el código del servidor ejecuta JavaScript en la página alojada y recibe un valor, el mecanismo que hay detrás de `Application.Call` y de la capa de interoperabilidad de los widgets. El segundo es el del shell: ese JavaScript llama al objeto host que inyectó el WebView, y el shell invoca la cámara de la plataforma. La respuesta vuelve por el mismo camino y se genera como un evento .NET normal que transporta una matriz de bytes.

Lo que diferencia esto de un clic normal es el tiempo. El técnico puede pasar cuarenta segundos encuadrando, así que el controlador no puede bloquearse; termina inmediatamente y la solicitud queda pendiente en `Application.Session` bajo un token que has generado. La respuesta llega en otra solicitud, así que el código que guarda los bytes y rellena el cuadro de imagen se ejecuta fuera del evento original, y sus cambios en los controles no llegarán solos al navegador: termina con `Application.Update`, exactamente igual que en el trabajo con `Application.StartTask`.

La cancelación y el error también son resultados, no excepciones: el usuario se echó atrás, la solicitud caducó, el shell se suspendió y se reanudó. Todos llegan por el mismo evento, y el token es lo que te permite descartar una respuesta que nadie está esperando.

```
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 pantalla, cuatro shells

FieldOps se distribuye a un navegador, a una PWA, a una ventana de escritorio WebView2 y a un shell móvil .NET MAUI, y la página de la orden de trabajo es la misma en los cuatro. Lo que muestra depende de una comprobación de capacidades que se hace una vez por sesión: la aplicación pregunta a la página si hay un objeto host y qué dice el shell que puede hacer, y esa respuesta se guarda en la sesión para que la lean todas las pantallas. Lo que no debes hacer es analizar el user agent: describe el navegador, no si esta compilación de tu shell expone una cámara.

La matriz tiene una fila por capacidad y una columna por shell. La cámara es nativa en el shell móvil y, en el navegador de un teléfono, es el control **Upload**, que el navegador completa directamente desde la aplicación de cámara; el shell de escritorio ofrece en su lugar un cuadro de diálogo de archivos. La ubicación es nativa en el móvil y, en los demás casos, es la API Geolocation del navegador a la que se llega mediante interoperabilidad con JavaScript. Las notificaciones proceden del sistema operativo en los dos shells y de web push en el navegador y la PWA. La biometría solo existe donde la plataforma tiene un autenticador, así que FieldOps la usa para volver a desbloquear una sesión, nunca como única forma de entrar.

Por tanto, Take Photo y **Upload** son dos formas de presentar una misma acción, y el resto de la pantalla nunca sabe cuál se ejecutó. Una capacidad que el shell no comunica simplemente no aparece: un botón desactivado sin explicación es peor que uno ausente, y uno que lanza una excepción al pulsarlo es peor que ambos.

El permiso en el momento de uso

Delante de cada función del dispositivo hay dos barreras: el permiso de la plataforma, cuando el sistema operativo o el navegador piden al usuario que permita la cámara, la ubicación o las notificaciones, y tu propia explicación, que va primero. FieldOps muestra una frase (la ubicación muestra la distancia al siguiente emplazamiento y registra la hora de llegada) y solo entonces dispara la solicitud de la plataforma.

El momento decide la respuesta. Una solicitud de ubicación en la pantalla de inicio de sesión, antes de que el técnico haya visto una orden de trabajo, recibe un «no», y en varias plataformas ese no es permanente; la misma solicitud lanzada por Navigate en un trabajo cuya dirección está en pantalla recibe un «sí» casi siempre. Por eso FieldOps pide las notificaciones después de la primera asignación de la sesión, y la cámara cuando se pulsa Take Photo.

- Concedido: continúa y recuerda la concesión durante esta sesión.
- Denegado una vez: mantén la acción y el motivo; el siguiente trabajo puede obtener otra respuesta.
- Denegado de forma permanente: ofrece la alternativa manual (adjuntar una foto existente, escribir la dirección) y muestra dónde está el ajuste.

El único comportamiento que garantiza una negativa permanente es volver a preguntar en bucle. Registra el resultado de cada solicitud por capacidad y por sesión, anótalo en el registro y trata una negativa como un estado normal de la pantalla, no como un error.

Llevar los bytes a casa

Una foto de un teléfono actual ocupa de cuatro a ocho megabytes, y un técnico con una conexión rural adjunta tres por trabajo. Redimensiona antes de que los bytes crucen el puente: la solicitud al shell incluye un límite para el lado más largo y una calidad JPEG, y la alternativa del navegador dibuja el archivo elegido en un canvas con el mismo límite y lo vuelve a codificar antes de que **Upload** lo envíe. Seis megabytes se convierten en unos trescientos kilobytes: la diferencia entre una aplicación utilizable y una inutilizable con poca cobertura.

En el servidor, transmite en lugar de almacenar en memoria: copia el flujo entrante al almacenamiento y guarda solo el identificador en el estado de la sesión, porque una matriz de bytes aparcada en un objeto de pantalla se multiplica por cada técnico simultáneo. Escribe el archivo fuera de la raíz web con un nombre aleatorio y sírvelo mediante un controlador que compruebe que esta sesión tiene derecho a esa orden de trabajo.

La firma sigue el mismo proceso con otro origen: una superficie de dibujo recoge los trazos, la imagen aplanada se exporta como PNG y se envía, y el servidor la guarda con el identificador de la orden de trabajo, el nombre escrito del firmante y una marca de tiempo generada por el servidor, nunca una hora o una identidad proporcionadas por el cliente. La ubicación es el contenido más pequeño y el más sensible: guarda juntas la latitud, la longitud, el radio de precisión y la hora del servidor, porque una posición con dos kilómetros de margen no demuestra la llegada, y captúrala solo tras una acción explícita.



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:

- ✓ Llamar a funciones del dispositivo desde el código del servidor a través del puente del shell y recibir los resultados como eventos.
- ✓ Comprobar las capacidades en tiempo de ejecución para que la misma pantalla funcione en un navegador, una PWA y un shell nativo.
- ✓ Solicitar los permisos en el momento adecuado y gestionar la denegación con elegancia.
- ✓ Llevar fotos, firmas y datos de ubicación al servidor de forma segura y eficiente.

Guía del laboratorio / examen

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



Laboratorio práctico

Añade funciones del dispositivo a FieldOps: una acción Take Photo que use la cámara del dispositivo en el shell nativo y recurra a un control Upload en el navegador, una captura de firma subida como imagen, una acción Navigate que lea la ubicación del dispositivo y abra la aplicación de mapas, una notificación push cuando se asigne una orden de trabajo, y una comprobación de capacidades que muestre u oculte cada acción según el shell.

Punto de partida

Abres FieldOps tal como lo dejaron los módulos 1 a 4: la lista de órdenes de trabajo, la página de detalle y el flujo de actualización del estado funcionan en un navegador normal, la PWA se instala y se actualiza sola, el shell de escritorio WebView2 aloja la aplicación en una ventana nativa y el shell .NET MAUI la ejecuta en Android e iOS. La página de detalle ya tiene un panel Photos, un panel Signature y un botón Navigate, todos ellos marcadores de posición sin funcionalidad, y los shells exponen un puente al que todavía no llama nada de la aplicación web.

Enfoque sugerido

- ✓ Escribe un servicio de capacidades que se ejecute una vez al iniciar la sesión: pregunta a la página alojada si está presente el objeto host del shell, registra en la sesión qué funciones comunica (cámara, ubicación, notificaciones, biometría) y exponlas a todas las pantallas mediante un único método.
- ✓ Implementa Take Photo como una única acción del servidor con dos vías detrás: en los shells, una solicitud al puente limitada a 1600 píxeles en el lado más largo y calidad JPEG 70, y en el navegador, un control **Upload** que redimensiona antes de enviar; guarda los bytes devueltos fuera de la raíz web y muestra la miniatura mediante un controlador autorizado.
- ✓ Añade el panel de firma sobre una superficie de dibujo en la página, captura los trazos, exporta el PNG aplanado y súbelo con el identificador de la orden de trabajo, el nombre escrito del firmante y una marca de tiempo generada por el servidor.
- ✓ Conecta Navigate a una solicitud de ubicación que muestre antes su motivo, guarda la latitud, la longitud, el radio de precisión y la hora del servidor en el registro de la visita, y después pasa el control a la aplicación de mapas de la plataforma a través del shell, o abre un enlace de mapas en una pestaña nueva del navegador cuando no hay puente.
- ✓ Genera una notificación cuando se asigne una orden de trabajo (a través del shell cuando el sistema operativo la ofrezca, y mediante web push en caso contrario), solicita el permiso de notificaciones después de la primera asignación de la sesión y no al iniciar sesión, y registra cada resultado de permiso por capacidad.

Entregables requeridos

- ✓ Captura con la cámara con alternativa de subida en el navegador

- ✓ Captura y subida de la firma
- ✓ Geolocalización y traspaso a la aplicación de mapas
- ✓ Notificación push al asignar
- ✓ Matriz de comprobación de capacidades entre shells

Preguntas de repaso

- ✓ El resultado de la cámara llega cuarenta segundos después del clic que lo pidió. Explica en qué solicitud llega, qué estado de sesión debe encontrar para saber qué hacer con los bytes y por qué el cuadro de imagen no aparece en el navegador sin un envío explícito.
- ✓ Hace tres semanas, un técnico denegó de forma permanente el permiso de ubicación en su propio teléfono. Describe exactamente qué debería hacer hoy la pulsación de Navigate y nombra lo único que la aplicación no debe hacer nunca en respuesta.
- ✓ Take Photo funciona en el shell móvil y la alternativa Upload funciona en Chrome de escritorio. Nombra algo que sea realmente distinto para el usuario entre esas dos vías y argumenta si esa diferencia es aceptable para un técnico de campo.



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.