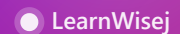


Configuración de una progressive web app

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Manifiesto de la aplicación web, iconos y pantallas de bienvenida, el service worker, la instalabilidad, el shell sin conexión y el flujo de actualización de una PWA. 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

Manifiesto de la aplicación web, iconos y pantallas de bienvenida, el service worker, la instalabilidad, el shell sin conexión y el flujo de actualización de una PWA.

1. El problema en producción

Los técnicos de FieldOps no abren un navegador y escriben una URL. Tocaban un icono en un teléfono que ha pasado toda la mañana en una furgoneta, en una sala de máquinas donde la señal desaparece a mitad de un trabajo. Una progressive web app les da ese icono, una ventana sin el marco del navegador y una página que sigue diciendo algo útil cuando desaparece la red; pero FieldOps es una aplicación Wisej.NET, así que su interfaz vive en el servidor, y fingir lo contrario es la forma en que una PWA se convierte en una incidencia de soporte.

2. Conceptos clave

Concepto	Por qué importa
manifiesto de la aplicación web	Un pequeño archivo JSON, enlazado desde la página de inicio, que indica al navegador el nombre de la aplicación, sus iconos, sus colores, su modo de visualización y su URL de inicio; sin él, el navegador no tiene nada que instalar.
modo de visualización	El miembro <code>display</code> del manifiesto. <code>standalone</code> abre FieldOps en su propia ventana sin barra de direcciones, que es lo que hace que una aplicación instalada deje de parecer un marcador.
iconos maskable	Iconos declarados con <code>purpose: "maskable"</code> y un margen de seguridad, para que Android pueda recortarlos con su propia forma sin cortar el dibujo.
service worker	Un script que el navegador ejecuta junto a la página, servido desde el propio origen de la aplicación por HTTPS, que puede responder a las solicitudes de la página desde una caché cuando la conexión es lenta o no existe.
shell de la aplicación	Los archivos estáticos que necesita una sesión antes de que el servidor pueda decir nada: la página de inicio, los scripts de cliente, el tema y los iconos. Es lo que el worker puede almacenar en caché.
primero caché frente a solo red	La regla de enrutamiento dentro del controlador fetch del worker: los archivos estáticos del shell se sirven primero desde la caché, mientras que todo lo que la sesión de Wisej.NET intercambia con el servidor debe ir directamente a la red, nunca a una copia en caché.
ciclo de vida de la actualización	Un worker nuevo se instala mientras el antiguo sigue controlando la página, y después espera. <code>skipWaiting</code> , <code>clients.claim</code> y una recarga controlada son lo que convierte ese worker en espera en la versión que el técnico ejecuta realmente.

3. Desarrollo de la lección

El manifiesto es el contrato de instalación

La instalabilidad no es un interruptor de Wisej.NET; es un conjunto de condiciones que comprueba el navegador. FieldOps debe servirse por HTTPS, su página de inicio debe enlazar un manifiesto de aplicación web válido y debe haber un service worker registrado para el ámbito de la página. El manifiesto está con los demás archivos estáticos de la aplicación (en la misma carpeta que la página de inicio, el tema y los iconos) y se referencia desde `Default.html`, que es el documento de inicio de Wisej.NET y el único lugar del proyecto donde controlas el HTML del navegador.

Los miembros que importan son pocos. `name` y `short_name` deciden lo que aparece bajo el icono y en el selector de tareas. `icons` necesita un PNG de 192 píxeles y otro de 512, más variantes maskable con el dibujo con margen, e iOS quiere además un enlace `apple-touch-icon` en la propia página. `theme_color` tiñe las barras del sistema alrededor de la ventana standalone, mientras que `background_color` pinta la pantalla de bienvenida que genera el navegador al iniciar, así que ambos deberían coincidir con la barra de la aplicación de FieldOps. `display` es `standalone`, y `start_url` con `scope` apuntan a la raíz de la

aplicación para que un inicio llegue a la lista de órdenes de trabajo y todas las URL de la aplicación se queden dentro de la ventana instalada.

Cuando el navegador queda satisfecho, dispara `beforeinstallprompt` en la página en lugar de mostrar su propio banner. El controlador de `Default.html` detiene ese comportamiento predeterminado, guarda el evento y expone una pequeña función que llama a su método `prompt()`. De ahí se derivan dos reglas. La invitación solo se abre con un clic real dentro del navegador, así que el botón que la dispara va en la página, no en el servidor. Y el servidor decide *cuándo* merece la pena hacer la oferta (el laboratorio la pide después de la primera sesión correcta), mostrando el botón mediante `Application.Eval` o la interoperabilidad con JavaScript equivalente:

```
private void OnFirstSessionCompleted()
{
    // Default.html stashed the beforeinstallprompt event; the server only
    // decides when the offer becomes visible. prompt() still runs from the
    // page's own click handler, inside a real user gesture.
    Application.Session["InstallOffered"] = true;
    Application.Eval("window.fieldOps.showInstallButton()");
}
```

Un worker que almacena en caché el shell, nunca la sesión

Registra el worker desde `Default.html` con `navigator.serviceWorker.register`, apuntando a un `sw.js` servido desde la raíz de la aplicación para que su ámbito cubra toda la aplicación. En `install`, el worker abre una caché con nombre y guarda en ella el shell: la página de inicio, los scripts de cliente y los archivos del tema que carga la página antes de que exista la sesión, el manifiesto, los iconos y la página sin conexión. En `activate` elimina todas las cachés cuyo nombre no sea el actual.

El controlador `fetch` es donde una PWA sobre una interfaz controlada por el servidor triunfa o fracasa. FieldOps es una aplicación Wisej.NET: los controles son objetos C# en el servidor, y lo que viaja durante una sesión es un flujo de solicitudes y respuestas que reflejan ese estado del servidor, más un WebSocket para las actualizaciones enviadas desde el servidor. Nada de eso se puede almacenar en caché con algún sentido: una respuesta reproducida describe una sesión que ya no existe. Por eso el controlador filtra según la solicitud: un `GET` de un archivo conocido del shell se responde primero desde la caché, las solicitudes que no son `GET` no se tocan nunca y todo lo demás va a la red sin tocarlo. Cuando una navegación falla porque el servidor no está disponible, el controlador responde con la página sin conexión en caché en lugar de la pantalla de error del navegador.

Un modo sin conexión honesto para una interfaz controlada por el servidor

Di claramente qué significa aquí sin conexión: el shell se carga, la aplicación no se ejecuta. Sin el servidor no hay sesión, ni datos de órdenes de trabajo, ni forma de cambiar un estado, y ninguna cantidad de caché traslada una interfaz del servidor al navegador. El trabajo de diseño consiste en que ese estado sea útil en lugar de parecer roto. FieldOps escribe en el almacenamiento del navegador un pequeño resumen tras cada carga correcta (número de órdenes de trabajo abiertas, siguiente emplazamiento y hora de la sincronización), y la página sin conexión lo muestra detrás de un aviso claro de «sin conexión» y un botón para reintentar.

- Nada de lo capturado en la página sin conexión se presenta como guardado; las modificaciones en cola son una función posterior y deliberada, no un efecto secundario de una caché.
- Todas las superficies sin conexión llevan la marca de tiempo de la última sincronización, para que los datos antiguos nunca se confundan con los actuales.
- Cuando vuelve la conectividad, la aplicación se recarga en una sesión nueva en lugar de reanudar una obsoleta.

La reconexión merece la misma honestidad. Una sesión de Wisej.NET puede caducar o perderse mientras el dispositivo está en reposo en una furgoneta; la recuperación fiable es recargar la página de inicio y dejar que el servidor construya una sesión nueva, no dejar en pantalla una interfaz muerta que parece activa.

Publicar una versión nueva a pesar de una caché que hace bien su trabajo

La caché que hace rápida a FieldOps es también la que deja a un técnico con la compilación del mes pasado. Pon una versión en el nombre de la caché (`fieldops-shell-v7`) y cámbiala en el mismo commit que cualquier archivo del shell. Así el worker nuevo se instala en una caché nueva y su controlador `activate` elimina las antiguas. Llama a `skipWaiting` para que el worker nuevo no espere detrás del antiguo, y a `clients.claim` para que controle inmediatamente la página abierta.

La página todavía tiene que reaccionar. Escuchar `controllerchange`, o `updatefound` y `statechange` en el registro, le indica que una versión nueva ha tomado el control; FieldOps muestra una pequeña barra «Hay una versión nueva — Recargar» y recarga cuando el técnico la toca, para que nadie pierda un formulario a medio rellenar por una actualización inesperada. Dos detalles de despliegue completan el flujo: `sw.js` y el manifiesto se sirven con cabeceras no-cache para que el navegador pueda detectar que han cambiado, y la aplicación compara su sello de compilación con el del servidor al iniciar la sesión, detectando un shell que sobrevivió a un despliegue en lugar de dejar que hable con un servidor con el que ya no coincide.



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:

- ✓ Añadir un manifiesto de aplicación web con iconos, colores y modo de visualización para que la aplicación se pueda instalar desde el navegador.
- ✓ Registrar un service worker que almacene en caché el shell de la aplicación y lo sirva cuando la red es lenta o no existe.
- ✓ Explicar qué puede y qué no puede funcionar sin conexión en una interfaz del lado del servidor y diseñar la experiencia sin conexión con honestidad.
- ✓ Gestionar las actualizaciones de la PWA para que los usuarios reciban la versión nueva sin una caché obsoleta.

Guía del laboratorio / examen

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



Laboratorio práctico

Convierte FieldOps en una PWA: añade el manifiesto con iconos, color del tema y visualización standalone, registra un service worker que almacene en caché el shell y los recursos estáticos, muestra una invitación a instalar después de la primera sesión correcta, muestra una página sin conexión con el último resumen sincronizado de las órdenes de trabajo cuando el servidor no esté disponible, e implementa el flujo de actualización que recarga la aplicación cuando hay una versión nueva.

Punto de partida

Abres la solución FieldOps que construiste en el Módulo 1: una aplicación web Wisej.NET con una lista de órdenes de trabajo, una página de detalle de la orden de trabajo y un flujo de actualización del estado que ya funcionan correctamente en una pestaña normal del navegador. La aplicación todavía no se puede instalar (no hay manifiesto, ni service worker, ni página sin conexión) y los archivos estáticos que vas a añadir estarán junto a los archivos `Default.html`, del tema y de los iconos que ya existen.

Enfoque sugerido

- ✓ Sirve FieldOps por HTTPS en local y después escribe `manifest.webmanifest` con el nombre de la aplicación, iconos de 192 y 512 píxeles con variantes maskable, los colores del tema y del fondo de FieldOps, `display: standalone` y una URL de inicio en la raíz de la aplicación, y enlázalo, junto con un apple-touch-icon, desde `Default.html`.
- ✓ Añade `sw.js` en la raíz de la aplicación con un nombre de caché con versión, un controlador install que precargue el shell (página de inicio, scripts de cliente, tema, iconos y la página sin conexión) y un controlador activate que elimine todas las cachés de versiones anteriores.
- ✓ Escribe el controlador fetch de modo que las solicitudes del shell se respondan primero desde la caché, mientras que todas las solicitudes que la sesión de Wisej.NET hace al servidor van a la red sin tocarlas, y una navegación fallida recurre a la página sin conexión en caché.
- ✓ Captura `beforeinstallprompt` en la página, guarda el evento y haz que el servidor muestre el botón Install de la página mediante interoperabilidad con JavaScript solo después de que el técnico haya completado una sesión correcta con la lista de órdenes de trabajo cargada.
- ✓ Escribe un resumen compacto de las órdenes de trabajo en el almacenamiento del navegador tras cada carga correcta para que la página sin conexión pueda mostrarlo con la marca de tiempo de la última sincronización; después incrementa la versión de la caché, despliega y confirma que aparece la barra de actualización y que recarga con la nueva compilación.

Entregables requeridos

- ✓ Manifiesto de la aplicación web con iconos y colores
- ✓ Service worker que almacena el shell en caché
- ✓ Invitación a instalar tras la primera sesión

- ✓ Página sin conexión con el último resumen sincronizado
- ✓ Flujo de actualización y comprobación de versión

Preguntas de repaso

- ✓ Un revisor pregunta por qué el service worker no almacena simplemente en caché todas las respuestas para que FieldOps siga funcionando bajo tierra. Explica qué haría realmente una respuesta reproducida de la sesión de Wisej.NET en la pantalla del técnico y dónde está el límite honesto.
- ✓ Tu oferta de instalación funciona en tu escritorio pero nunca aparece en el teléfono del técnico. Nombra tres condiciones distintas que podrían ser la causa y di cómo las distinguirías desde el dispositivo.
- ✓ Un técnico tiene la aplicación instalada y abierta cuando despliegas una nueva compilación. Describe todos los pasos entre tu despliegue y ese teléfono ejecutando el nuevo shell, e identifica qué paso fallaría en silencio si el nombre de la caché no cambiara nunca.



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.