Engineer

Aplicación web progresiva

sitio web que puede instalarse y usarse como una aplicación

Qué es

Una aplicación web progresiva, también escrita Progressive Web App y normalmente abreviada como PWA, es un sitio web construido para que un navegador pueda instalarlo y ejecutarlo como una aplicación: desde su propio icono, en su propia ventana y, a menudo, con una parte que sigue funcionando cuando no hay red. En todo momento sigue siendo un sitio web corriente. No procede de una tienda de aplicaciones, y las páginas se componen del mismo HTML, CSS y JavaScript que un navegador cargaría en una pestaña.

Ningún estándar define el término. Lo que definen los estándares son las dos partes que lo componen. La primera es el manifiesto de aplicación web, un archivo JSON descrito por Web Application Manifest del W3C, que da a la aplicación un nombre, iconos, una URL de inicio y un modo de visualización. Esa especificación establece un umbral bajo para cumplir los requisitos: «Cualquier sitio web es una aplicación web instalable». Un navegador puede ofrecer una forma de instalarlo, y la aplicación instalada se abre después con los ajustes del manifiesto aplicados. En el modo de visualización `standalone` se pretende que se parezca a una aplicación nativa, lo que puede significar una ventana propia y un icono en el lanzador de aplicaciones, y el navegador omite su interfaz habitual, como la barra de URL.

La segunda parte es el service worker, definido por la especificación Service Workers del W3C. Es un script que el navegador ejecuta separado de la página. Se sitúa entre esta y la red, ve las solicitudes que hace la página y puede responderles desde cachés que controla el script. Eso permite que una aplicación web siga sirviendo sus páginas sin conexión. Esas cachés pertenecen al script: la especificación las mantiene completamente separadas de la caché HTTP ordinaria del navegador, y solo cambian cuando el script las modifica.

Web Application Manifest nunca usa la palabra «progressive», y la especificación Service Workers la usa una vez, en una nota sobre cómo Chromium reconoce una PWA. La guía de Next.js sobre PWA recomienda la mejora progresiva para que la aplicación siga funcionando bien en un navegador que no admita algunas de sus funciones de PWA. En ese sentido, cada parte es opcional. Un navegador que ignora el manifiesto o no admite service workers sigue mostrando el sitio web.

Una aplicación web progresiva tampoco es una clase distinta de sitio web. Una aplicación de una sola página, un sitio estático y uno renderizado en el servidor pueden añadir un manifiesto y un service worker. El término describe lo que un navegador puede hacer con un sitio después de cargarlo, no cómo se produce su HTML.

Utilice el concepto en una decisión

Aplique «¿Qué es una aplicación web progresiva?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Interfaces que cargan rápido, funcionan en el teléfono y se comportan igual en los navegadores que utilizan sus clientes.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Dirección del desarrollo integral de mapas GIS — cartografía integral, de la base de datos al navegador

Dirección del desarrollo integral de mapas GIS Póngase en contacto →

Por qué importa

La guía de Next.js defiende la propuesta con tres argumentos. Las actualizaciones llegan de inmediato, sin esperar la aprobación de una tienda de aplicaciones. Una sola base de código sirve para todas las plataformas. Y la aplicación obtiene funciones que se esperan de las aplicaciones nativas, como la instalación en la pantalla de inicio y las notificaciones push. Los tres se derivan de que la aplicación siga siendo un sitio web: se puede publicar una nueva versión sin un lanzamiento independiente en una tienda de aplicaciones; cuándo la adopta cada dispositivo depende de la caché y de las actualizaciones del service worker.

El funcionamiento sin conexión es la razón de ser del service worker. La sección de motivación de la propia especificación dice que las aplicaciones web han supuesto tradicionalmente que se puede acceder a la red y que, por ello, otras plataformas de aplicaciones resuelven mejor las conexiones interrumpidas. Los service workers dan a las aplicaciones web una forma de gestionar también esa

interrupción. Con uno, quien viaje en un tren dentro de un túnel o en un buque fuera del alcance de la costa ya no tiene por qué acabar ante la página de error del navegador.

Las dos partes son independientes. La guía de Next.js señala que los avisos de instalación no necesitan soporte sin conexión, y la especificación del manifiesto ha eliminado su propio miembro `serviceworker`. Una aplicación puede ser instalable sin service worker, y un service worker puede hacer que un sitio corriente funcione sin conexión sin ningún manifiesto.

Lo que ambas partes comparten es HTTPS. Next.js enumera un sitio servido por HTTPS como una de las dos cosas que una aplicación necesita para instalarse en la pantalla de inicio de un móvil; la otra es un manifiesto válido. La especificación Service Workers dice que los service workers deben ejecutarse en contextos seguros, lo que en la práctica significa HTTPS, con localhost permitido para el desarrollo.

Cómo se aplica en la práctica

Tres archivos pequeños forman una PWA funcional, todos escritos para este artículo. El manifiesto da nombre a la aplicación, elige el modo de visualización y enumera sus iconos:

```
{
  "name": "Field Notes",
  "short_name": "Notes",
  "start_url": "/",
  "display": "standalone",
  "icons": [
    { "src": "/icon-192.png", "sizes": "192x192", "type": "image/png" },
    { "src": "/icon-512.png", "sizes": "512x512", "type": "image/png" }
  ]
}
```

Cada página enlaza el manifiesto y registra el service worker, pero solo en un navegador que lo admita:

```
<link rel="manifest" href="/manifest.webmanifest" />
<script>
  if ("serviceWorker" in navigator) {
```

```
    navigator.serviceWorker.register("/sw.js");
  }
</script>
```

Y el service worker conserva un pequeño conjunto de archivos para usarlos sin conexión:

```
// sw.js
const CACHE = "shell-v2";
const SHELL = ["/offline.html", "/app.css", "/app.js"];

async function precache() {
  const cache = await caches.open(CACHE);
  await cache.addAll(SHELL);
}

async function dropOldCaches() {
  const names = await caches.keys();
  const old = names.filter((name) => name !== CACHE);
  await Promise.all(old.map((name) => caches.delete(name)));
}

async function respond(request) {
  if (request.mode === "navigate") {
    try {
      return await fetch(request);
    } catch {
      return caches.match("/offline.html");
    }
  }
  const hit = await caches.match(request);
  return hit ?? fetch(request);
}

self.addEventListener("install", (event) => {
  event.waitUntil(precache());
});

self.addEventListener("activate", (event) => {
  event.waitUntil(dropOldCaches());
});

self.addEventListener("fetch", (event) => {
  if (event.request.method !== "GET") return;
  event.respondWith(respond(event.request));
});
```

El manifiesto usa `.webmanifest`, la extensión de archivo registrada para él, y se sirve con el tipo de medio `application/manifest+json`. El service worker se sitúa en la raíz porque su ámbito predeterminado es el directorio donde está el script, por lo que uno ubicado en `/js/sw.js` solo podría controlar páginas bajo `/js/` salvo que el servidor enviara una cabecera `Service-Worker-Allowed`. Al instalarse, abre una caché llamada `shell-v2` y guarda la página sin conexión, la hoja de estilo y el script. `addAll()` rechaza la operación si alguno de esos archivos devuelve un estado de error y, como esa promesa se pasó a `waitUntil()`, toda la instalación falla con ella en lugar de dejar una caché a medio llenar. Al activarse, elimina todas las cachés con otro nombre. Las navegaciones de página acuden primero a la red y recurren a la página sin conexión almacenada en caché cuando la red no responde. Todo lo demás se sirve desde la caché cuando está allí y se obtiene de la red cuando no está. El service worker no interviene en las solicitudes distintas de `GET`.

La empresa describe una implementación completa en una plataforma real. La entrada del portafolio explica por qué hacía falta: muchos usuarios de la plataforma son gente de mar y personal de campo que utiliza teléfonos, a menudo sin conexión o con una señal débil, por lo que la aplicación debía instalarse como una nativa y seguir funcionando cuando desapareciera la señal. Cubre ambas partes. La aplicación se instala en la pantalla de inicio con iconos de varios tamaños, una pantalla de presentación y colores del tema. El service worker se configura mediante el complemento next-pwa, y el almacenamiento en caché de Workbox durante la ejecución sirve fuentes, imágenes, audio, vídeo y recursos de CDN dando prioridad a la caché, porque ninguno cambia de una solicitud a la siguiente, junto con cabeceras HTTP de control de caché. Como los service workers necesitan HTTPS, se probaron la instalación y el comportamiento sin conexión por HTTPS en dispositivos reales. El resultado que declara es una aplicación que las personas instalan y siguen usando sin conexión, con visitas posteriores que cargan desde la caché.

La parte de instalación también cambia desde el punto de vista de quien usa la aplicación. La página de este sitio sobre cómo instalarlo como aplicación explica dónde guarda cada navegador el comando: el menú Compartir y Añadir a pantalla de inicio en Safari en un iPhone; un pequeño icono a la derecha de la barra de direcciones en Chrome, Edge, Brave, Opera y Vivaldi en un ordenador; y Añadir al Dock en Safari en macOS Sonoma o posterior. También afirma que iOS nunca sugiere la instalación por iniciativa propia. Firefox ofrece ventanas independientes para aplicaciones web en Windows desde la versión 143, o desde la 150 si Firefox se ha instalado desde Microsoft Store, pero no en macOS ni

Linux. La página también deja claro qué aporta instalar este sitio: un icono y una ventana, pero ninguna copia sin conexión, porque ese soporte necesita un service worker.

Dónde surgen los problemas

Los usuarios conservan la versión antigua. Según la especificación Service Workers, los objetos de caché no se actualizan salvo que el autor lo solicite expresamente y no caducan salvo que este los elimine. Por tanto, un service worker que da prioridad a la caché sirve la hoja de estilo de ayer hasta que su script diga otra cosa. Incluya una versión en el nombre de la caché, cámbiela cuando cambien los archivos almacenados y elimine las cachés antiguas al activarse, como hace el ejemplo. Cambiar el script es lo que inicia una actualización, porque el navegador compara el nuevo `sw.js` descargado, byte a byte, con el instalado. Incluso entonces, el nuevo service worker espera hasta que ninguna pestaña ni ventana use el anterior, salvo que invoque `skipWaiting()` , de modo que un usuario que mantiene una pestaña abierta ejecuta la versión anterior mientras siga abierta. La guía de Next.js sirve `sw.js` con `Cache-Control: no-cache, no-store, must-revalidate` para que los usuarios reciban siempre la versión más reciente del propio service worker.

Funciona en el portátil, pero no en el teléfono de pruebas. `navigator.serviceWorker` solo existe en contextos seguros. En una página servida por HTTP sin cifrar, la comprobación habitual de compatibilidad devuelve falso sin avisar, el service worker nunca se registra y nada en la página lo indica. La especificación permite que los navegadores admitan localhost, 127.0.0.0/8 y ::1/128 para el desarrollo; por eso un service worker que funciona en la máquina del desarrollador puede no estar disponible cuando un teléfono abre la misma aplicación por HTTP desde la dirección de esa máquina en la red local. Next.js ofrece `next dev --experimental-https` para pruebas locales por HTTPS. Pruebe la instalación y el comportamiento sin conexión en dispositivos reales y por HTTPS antes de dar cualquiera de los dos por terminado.

Un manejador de solicitudes que no hace nada. La especificación Service Workers recoge que algunos sitios registran un manejador de `fetch` con el cuerpo vacío solo para que Chromium los reconozca como aplicaciones web progresivas. La especificación detecta ahora manejadores como `() => {}` y anima a los navegadores a advertir que son innecesarios y pueden perjudicar el rendimiento. Escriba un manejador que responda a las solicitudes o no lo añada.

Dar por hecho que aparecerá un aviso de instalación. Decidir si se ofrece la instalación y de qué manera corresponde al navegador. Next.js dice que los navegadores modernos muestran automáticamente un aviso cuando el sitio tiene un manifiesto válido y utiliza HTTPS, y desaconseja crear un botón de instalación propio basado en `beforeinstallprompt` porque ese evento no funciona en todos los navegadores y plataformas, incluido Safari en iOS. La especificación del manifiesto ha eliminado después el evento por completo. El modo de visualización y la URL de inicio también son orientativos: un navegador que no admite `standalone` recurre a `minimal-ui` y después a `browser`, una pestaña corriente. Explique cómo instalar, como hacen las instrucciones para iOS de la guía de Next.js, y diseñe contando con que la aplicación pueda abrirse en una pestaña.

Lecturas adicionales

- Web Application Manifest — El formato del manifiesto, sus miembros, los modos de visualización y qué significan la instalación y la desinstalación.
- Service Workers — El ciclo de vida del service worker, su ámbito, las reglas de actualización, el requisito de contexto seguro y la Cache API.
- Web Application Manifest: aplicaciones web instalables — Cómo un navegador instala una aplicación web, la inicia y permite que el usuario vuelva a eliminarla.
- Service Workers: duración de las cachés — Por qué los objetos de caché nunca se actualizan ni caducan salvo que el script lo indique.
- Cómo crear una aplicación web progresiva (PWA) con Next.js — Un archivo de manifiesto, notificaciones push, instrucciones de instalación para iOS, HTTPS en desarrollo y cabeceras de seguridad para el service worker.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una aplicación web progresiva?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/progressive-web-app/