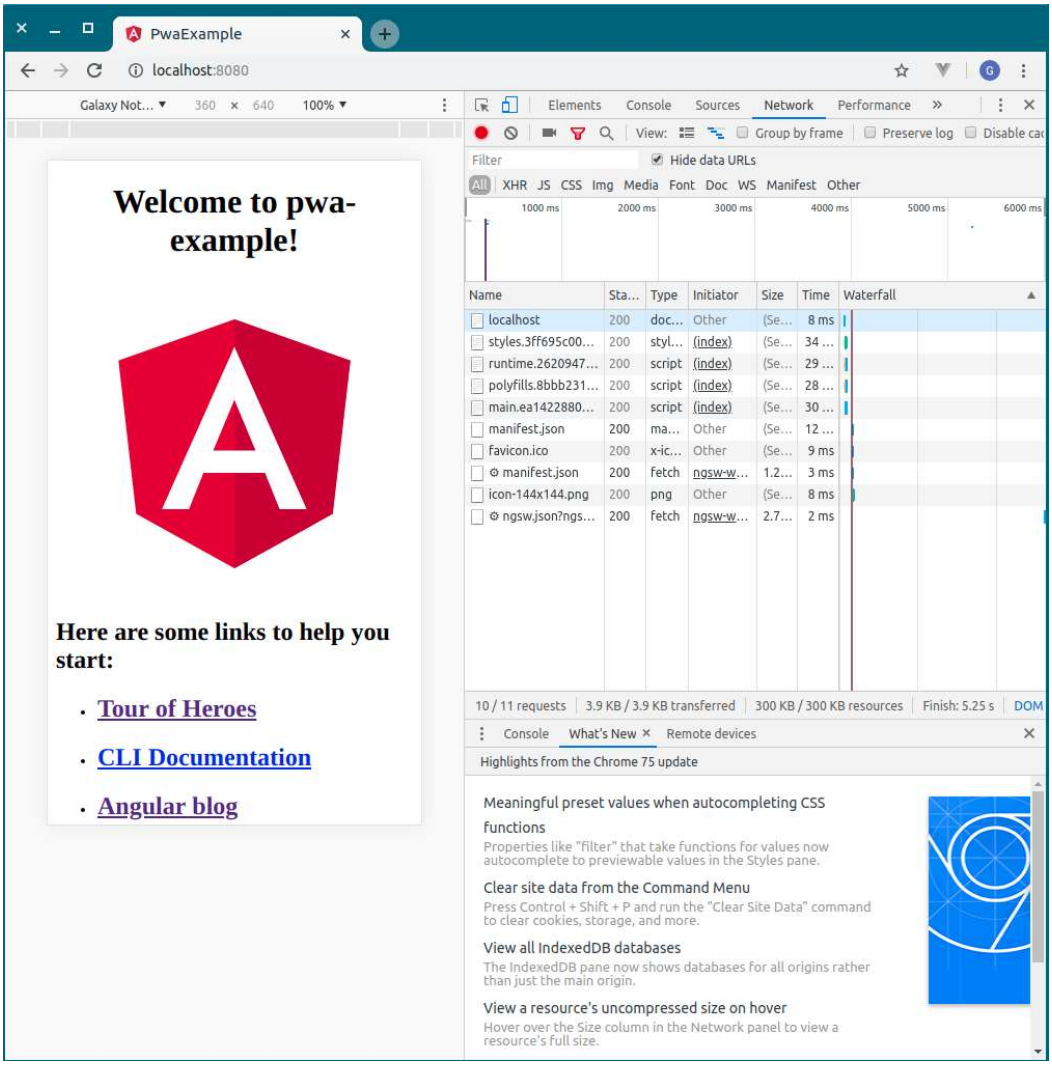
Introducción a aplicaciones web progresivas

Este articulo provee una introducción a las Aplicaciones web progresivas (PWAs), explica qué son, y las ventajas que brindan sobre las aplicaciones web convencionales.



¿Qué es una aplicación web progresiva?

El término "Aplicación web progresiva" no es un nombre formal u oficial. Solo es una abreviatura utilizada inicialmente por Google para el concepto de crear una aplicación flexible y adaptable utilizando solo tecnologías web.
Las PWA son aplicaciones web desarrolladas con una serie de tecnologías específicas y patrones estándar que les permiten aprovechar las funciones de las aplicaciones nativas y web. Por ejemplo, las aplicaciones web son más fáciles de detectar que las aplicaciones nativas; es mucho más fácil y rápido visitar un sitio web que instalar una aplicación, y también puedes compartir aplicaciones web simplemente enviando un enlace.

Por otro lado, las aplicaciones nativas están mejor integradas con el sistema operativo y, por lo tanto, ofrecen una experiencia más fluida para los usuarios. Puedes instalar una aplicación nativa para que funcione sin conexión, y a los usuarios les encanta tocar sus íconos para acceder fácilmente a sus aplicaciones favoritas, en lugar de navegar a través de un navegador.

Las PWA brindan la capacidad de crear aplicaciones web que pueden disfrutar de estas mismas ventajas.

No es un concepto completamente nuevo; estas ideas se han revisado muchas veces en la plataforma web con varios enfoques en el pasado. La mejora progresiva y el diseño adaptable ya te permiten crear sitios web compatibles con dispositivos móviles.

Sin embargo, las PWA brindan todo esto y más sin perder ninguna de las características existentes que hacen que la web sea excelente.

¿Qué hace que una aplicación sea una PWA?

Como dijimos anteriormente, las PWA no se crean con una sola tecnología. Representan una nueva filosofía para la creación de aplicaciones web, que incluye algunos patrones específicos, API y otras características. A primera vista, *no* es tan obvio si una aplicación web es una PWA o no. Una aplicación se podría considerar una PWA cuando cumple con ciertos requisitos o implementa un conjunto de características determinadas — funciona sin conexión, es instalable, es fácil de sincronizar, puede enviar notificaciones automáticas, etc.

Además, existen herramientas para medir qué tan completa (como porcentaje) es una aplicación web, como Lighthouse. Al implementar varias ventajas tecnológicas, podemos hacer que una aplicación sea más progresiva, y así terminar con una puntuación de Lighthouse más alta. Pero este es solo un indicador aproximado.

Hay algunos principios clave que una aplicación web debe tratar de observar para ser identificada como PWA. Estos deben ser:

- Detectable (en-US), por lo que el contenido se puede encontrar a través de motores de búsqueda.
- Instalable (en-US), por lo que puede estar disponible en la pantalla de inicio del dispositivo o en el lanzador de aplicaciones.
- Enlazable (en-US), para que puedas compartirla simplemente enviando una URL.
- Independiente de la red (en-US), por lo que funciona sin conexión o con una deficiente conexión de red.
- Progresiva (en-US), por lo que todavía se puede utilizar en un nivel básico en los navegadores más antiguos, pero completamente funcional en los más recientes.

- Reconectable (en-US), por lo que puede enviar notificaciones cuando haya contenido nuevo disponible.
- Adaptable (en-US), por lo tanto se puede utilizar en cualquier dispositivo con pantalla y navegador: teléfonos móviles, tabletas, computadoras portátiles, televisores, refrigeradores, etc.
- Segura (en-US) por lo que las conexiones entre el usuario, la aplicación y tu servidor están protegidos contra terceros que intenten acceder a datos sensibles.

Ofrecer estas funciones y hacer uso de todas las Ventajas que ofrecen las aplicaciones web puede crear una oferta atractiva y altamente flexible para tus usuarios y clientes.

¿Vale la pena hacer todo eso?

¡Absolutamente! Con un esfuerzo relativamente pequeño para implementar las características principales de las PWAs, los beneficios son enormes. Por ejemplo:

- Una disminución en los tiempos de carga después de la instalación de la aplicación, gracias al almacenamiento en caché con el servicio workers, además de ahorrar un valioso ancho de banda y tiempo. Los PWAs tienen una carga casi instantánea (a partir de la segunda visita).
- La capacidad de actualizar solo el contenido que ha cambiado cuando hay disponible una actualización de la aplicación. En contraste, con una aplicación nativa, incluso la más mínima modificación puede obligar al usuario a descargar la aplicación completa nuevamente.
- Una apariencia que está más integrada con la plataforma nativa: íconos de aplicaciones en la pantalla de inicio o el lanzador de aplicaciones, aplicaciones que se ejecutan automáticamente en modo de pantalla completa, etc.
- Reconectable para interactuar con los usuarios mediante el uso de notificaciones del sistema y mensajes push, lo cual genera usuarios más comprometidos y mejores tasas de conversión.

Historias de éxito

Hay muchas historias de éxito de empresas que probaron la ruta PWA, optaron por una experiencia de sitio web mejorada en lugar de una aplicación nativa y, como resultado, obtuvieron importantes beneficios medibles. El sitio web Estadísticas PWA comparte muchos estudios de casos que indican estos beneficios.

La historia de éxito más conocida probablemente es la de Flipkart Lite. El sitio de comercio electrónico más grande de la India se reconstruyó como una aplicación web progresiva en 2015, lo que resultó en un aumento del 70% en las conversiones. La PWA AliExpress también ha obtenido resultados mucho mejores que la web o la aplicación nativa, con un aumento del 104% en las tasas de conversión para los nuevos usuarios. Dado el aumento de sus ganancias y la cantidad relativamente baja de trabajo requerida para la conversión de estas aplicaciones a PWA, la ventaja es clara.

Las puestas en marcha emergentes en etapa temprana como couponmoto también han comenzado a usar aplicaciones web progresivas para impulsar una mayor participación de los consumidores, lo que demuestra que pueden ayudar tanto a pequeñas como a grandes empresas para (re)involucrar a los usuarios de manera más eficaz.

Puedes consultar la lista en pwa.rocks para obtener más ejemplos. Vale la pena mencionar en particular la página hnpwa.com, que muestra una implementación de ejemplo del sitio web de Hacker News (en lugar de la aplicación habitual TodoMVC), en la que puedes ver el uso de varios marcos de desarrollo de la interfaz de usuario web.

Incluso puedes generar PWA en línea utilizando el sitio web PWABuilder.

Para obtener información específica sobre el servicio *worker* y la inserción, asegúrate de consultar el Libro de recetas del servicio *worker*, una colección de recetas que utilizan los servicios *worker* en sitios modernos.

Vale la pena probar un enfoque de PWA, para que puedas ver por ti mismo si funciona para tu aplicación.

Ventajas de las aplicaciones web

Una aplicación web progresiva totalmente capaz debería proporcionar todas las siguientes ventajas al usuario.

Reconocible

El objetivo final es que las aplicaciones web tengan una mejor representación en los motores de búsqueda, sean más fáciles de exponer, catalogar y clasificar, y tener metadatos utilizables por los navegadores para brindarles capacidades especiales.

Algunas de las capacidades ya se han habilitado en ciertas plataformas basadas en web mediante tecnologías patentadas como Open Graph, que proporciona un formato para especificar metadatos similares en el bloque `<head>` de HTML con etiquetas `<meta>`.

El estándar web relevante aquí es el manifiesto de la aplicación web, que define las características de una aplicación, como el nombre, el icono, la pantalla de presentación y los colores del tema en un archivo de manifiesto con formato JSON. Esto es para usar en contextos como listas de aplicaciones y pantallas de inicio de dispositivos.

Instalable

Una parte fundamental de la experiencia de la aplicación web es que los usuarios tengan iconos de aplicaciones en su pantalla de inicio y los puedan tocar para abrir aplicaciones en su propio contenedor nativo que se sienta bien integrado con la plataforma subyacente.

Las aplicaciones web modernas pueden hacer que esta aplicación nativa se sienta a través de las propiedades establecidas dentro del manifiesto de la aplicación web y mediante una función disponible en los navegadores de teléfonos inteligentes modernos llamada instalación de la aplicación web.

Enlazable

Una de las características más poderosas de la web es la capacidad de vincularse a una aplicación en una URL específica sin la necesidad de una tienda de aplicaciones o un proceso de instalación complejo. Así ha sido siempre.

Independiente de la red

Las aplicaciones web modernas pueden funcionar cuando hay mala (o incluso inexistente) conectividad con la red. Las ideas básicas detrás de la independencia de la red son poder:

- Volver a visitar un sitio y obtener su contenido incluso si no hay una red disponible.
- Explorar cualquier tipo de contenido que el usuario haya visitado anteriormente al menos una vez, incluso en situaciones de mala conectividad.
- Controlar lo que se muestra al usuario en situaciones donde no hay conectividad.

Esto se consigue mediante una combinación de tecnologías: el servicio Workers para controlar las solicitudes de página (por ejemplo, almacenarlas sin conexión), la API de caché (en-US) para almacenar respuestas a solicitudes de red sin conexión (muy útil para almacenar activos del sitio) y tecnologías de almacenamiento de datos secundarios como Almacenamiento Web y IndexedDB para almacenar datos de aplicaciones sin conexión.

Compatibilidad de mejora progresiva

Se pueden desarrollar aplicaciones web modernas para proporcionar una experiencia excelente a los navegadores totalmente compatibles y una experiencia aceptable (aunque no tan brillante) a los navegadores menos capaces. Hemos estado haciendo esto durante años con las mejores prácticas, como la mejora progresiva. Al utilizar Mejora

progresiva, las PWAs se utilizan en varios navegadores. Esto significa que los desarrolladores deben tener en cuenta las diferencias en la implementación de algunas características y tecnologías PWA entre diferentes implementaciones de navegadores.

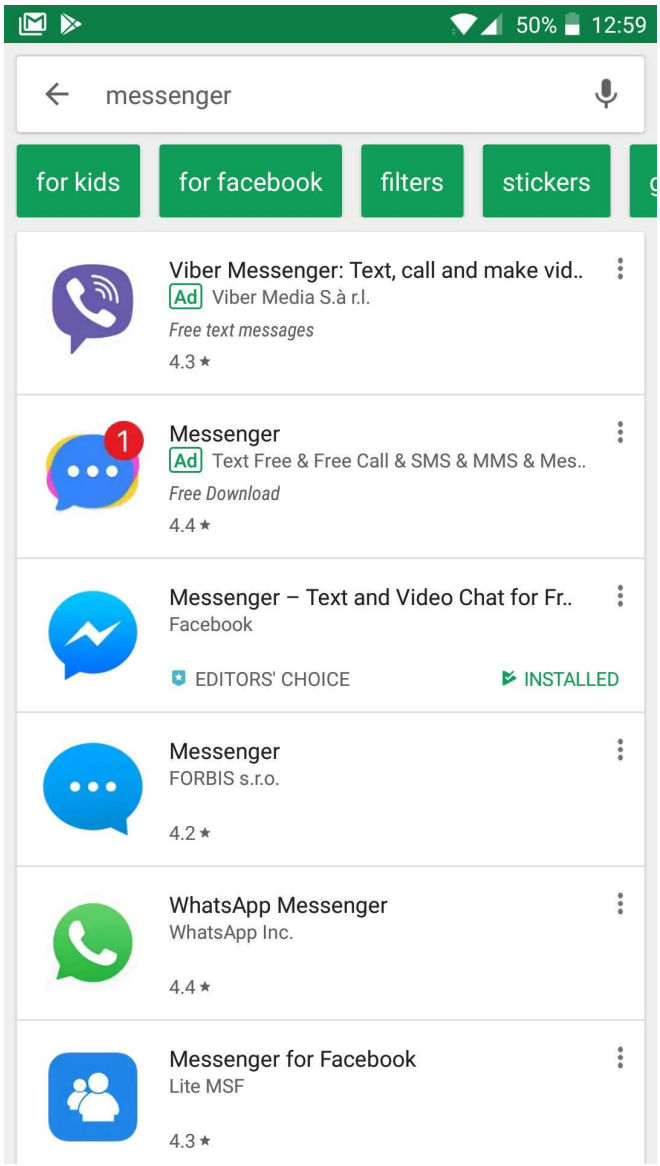
Reconectable

Una de las principales ventajas de las plataformas nativas es la facilidad con la que los usuarios pueden volver a interactuar con las actualizaciones y el contenido nuevo, incluso cuando no están mirando la aplicación o usando sus dispositivos. Las aplicaciones web modernas ahora también pueden hacer esto, utilizando nuevas tecnologías como Servicio *Workers* para controlar páginas, la API Web Push para enviar actualizaciones. directamente del servidor a la aplicación a través de un servicio *workers* y la API de notificaciones para generar notificaciones del sistema para ayudar a involucrar a los usuarios cuando no están utilizando activamente su navegador web.

Adaptable

Las aplicaciones web adaptables utilizan tecnologías como consultas de medios y viewport para asegurarte de que tu IU se ajuste a cualquier factor de forma: computadora de escritorio, dispositivo móvil, tableta o lo que venga a continuación.

Segura



La plataforma web proporciona un mecanismo de entrega seguro que evita espionaje y, al mismo tiempo, garantiza que el contenido no haya sido manipulado, siempre que aproveche HTTPS y desarrolles tus aplicaciones pensando en la seguridad.

También es fácil para los usuarios asegurarse de que están instalando la aplicación correcta, porque su URL coincidirá con el dominio de tu sitio. Esto es muy diferente de las aplicaciones en las tiendas de aplicaciones, que pueden tener varias aplicaciones con nombres similares, algunas de las cuales incluso pueden estar basadas en su propio sitio, lo que aumenta la confusión. Las aplicaciones web eliminan esa confusión y garantizan que los usuarios obtengan la mejor experiencia posible.

Compatibilidad con el navegador

Como se mencionó anteriormente, las PWAs no dependen de una sola API, sino que utilizan varias tecnologías para lograr el objetivo de brindar la mejor experiencia web posible.

El ingrediente clave requerido para las PWAs es la asistencia de servicio workers. Afortunadamente, los servicios de *workers* ahora son compatibles con los principales navegadores en computadoras de escritorio y dispositivos móviles.

Otras características como manifiesto de App Web, Push, Notificaciones y la funcionalidad Agregar a la pantalla de inicio (en-US) también tienen un amplio soporte. Actualmente, Safari tiene soporte limitado para el manifiesto de aplicaciones web y Agregar a la pantalla de inicio y no admite notificaciones *push web*. Sin embargo, otros navegadores importantes admiten todas estas funciones.

Por encima de todo, debes seguir la regla de mejora progresiva: usa tecnologías que mejoren la apariencia y la utilidad de tu aplicación cuando estén disponibles, pero que sigan ofreciendo la funcionalidad básica de tu aplicación cuando esas funciones no estén disponibles. Presentar un sitio web confiable con un buen rendimiento es una consecuencia del uso de estas mejoras; esto, a su vez, significa crear aplicaciones web que sigan las mejores prácticas. De esta manera, todos podrán usar la aplicación, pero aquellos con navegadores modernos se beneficiarán aún más de las funciones de PWA.

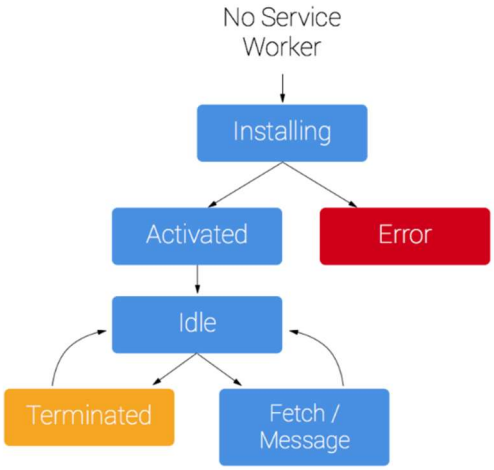
El servicio *workers* explicado

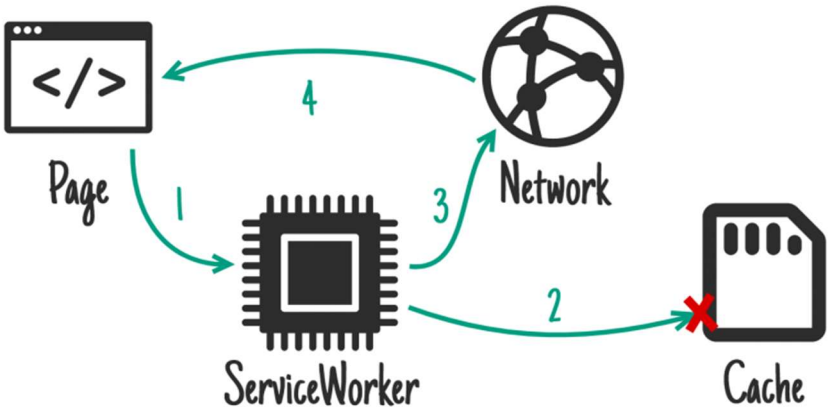
El servicio *workers* son un delegado virtual entre el navegador y la red. Finalmente, solucionan problemas con los que los desarrolladores de aplicaciones para el usuario han luchado durante años, en particular, cómo almacenar correctamente en caché los activos de un sitio web y ponerlos a disposición cuando el dispositivo del usuario está desconectado.

Se ejecutan en un hilo separado del código JavaScript principal de nuestra página y no tienen acceso a la estructura DOM. Esto introduce un enfoque diferente al de la programación web tradicional — la API no bloquea y puede enviar y recibir comunicación entre diferentes contextos. Puede darle a un servicio *worker* algo en lo que trabajar y recibir el resultado cuando esté listo utilizando un enfoque basado en una promesa.

Pueden hacer mucho más que "solo" ofrecer capacidades sin conexión, incluido el manejo de notificaciones, la realización de cálculos pesadosen un hilo separado, etc. El servicio *workers* es bastante poderoso, ya que pueden tomar el control de las solicitudes de red, modificarlas, entregar respuestas personalizadas recuperadas de la caché o sintetizar respuestas por completo.

Ciclo de vida de un service worker





Al iniciar el serviceworker se debe registrar, para ello realiza la **instalación**, aquí es donde se puede almacenar los archivos estáticos en la caché.

Después de la instalación, comenzará el paso de **activación**. Aquí podremos administrar los cachés anteriores. En la siguiente imagen podemos ver el proceso de activación, en él hemos programado un filtro para que coja las cachés antiguas y borre todas las cachés que empiecen por “marvel-” y que no sean la versión actual.

Siguiendo al proceso de **activación**, está el proceso **fetch**, donde el service worker controlará todas las páginas que están a su alcance.

En el proceso **fetch** hemos optado por programar **offline-first**, esto es que primero miramos la caché (2), y se carga lo que tengamos (lo que se ha guardado en el proceso de instalación), después vamos a Internet a por nueva información (3), y se descarga (4), finalmente pintamos en el navegador toda la información

Seguridad

Debido a que son tan poderosos, los Servicios *Workers* solo se pueden ejecutar en contextos seguros (es decir, HTTPS). Si deseas experimentar primero antes de enviar tu código a producción, siempre puedes probar en un anfitrión local o configurar las páginas de GitHub — ambas admiten HTTPS.

Desconectado primero

El patrón "desconectado primero" o "primero caché" es la estrategia más popular para entregar contenido al usuario. Si un recurso está almacenado en caché y disponible sin conexión, devuélvelo primero antes de intentar descargarlo del servidor. Si aún no está en la caché, descárgalo y almacénalo para uso futuro.

"Progresiva" en PWA

Cuando se implementa correctamente como una mejora progresiva, el servicio *workers* puede beneficiar a los usuarios que tienen navegadores modernos que admiten la API al brindar soporte fuera de línea, pero no romperán nada para aquellos que usan navegadores heredados.

Adding a service worker to your project

Para configurar el Service Worker de Angular en tu proyecto, utiliza el comando CLI ng add @angular/pwa. Se encarga de configurar tu aplicación para que utilice service workers añadiendo el paquete @angular/service-worker junto con la configuración de los archivos de soporte necesarios.

```
ng add @angular/pwa --project *project-name*
```

El comando anterior completa las siguientes acciones:

1. Añade el paquete @angular/service Worker a tu proyecto.
2. Activa el soporte de construcción de Service Worker en la CLI.
3. Importa y registra el Service Worker en el módulo de la aplicación.
4. Actualiza el archivo index.html:
   - o Incluye un enlace para añadir el archivo manifest.webmanifest.
   - o Añade una etiqueta meta para el theme-color.
5. Instala los archivos de iconos para dar soporte a la Progressive Web App (PWA) instalada.
6. Crea el archivo de configuración del Service Worker llamado ngsw-config.json, que especifica los comportamientos de caché y otros ajustes.

Ahora, construye el proyecto:

```
ng build
```

The CLI project is now set up to use the Angular service worker.

Service worker in action: a tour

Esta sección demuestra un Service Worker en acción, utilizando una aplicación de ejemplo..

Serving with http-server

Dado que ng serve no funciona con service workers, debe utilizar un servidor HTTP independiente para probar su proyecto localmente. Utiliza cualquier servidor HTTP. El siguiente ejemplo utiliza el paquete http-server de npm. Para reducir la posibilidad de conflictos y evitar servir contenido obsoleto, pruebe en un puerto dedicado y desactive el almacenamiento en caché.

Para servir el directorio que contiene tus archivos web con http-server, ejecuta el siguiente comando:

```
http-server -p 8080 -c-1 dist/<project-name>
```

Initial load

Con el servidor en funcionamiento, apunta al navegador a http://localhost:8080/. La aplicación debería cargarse normalmente.

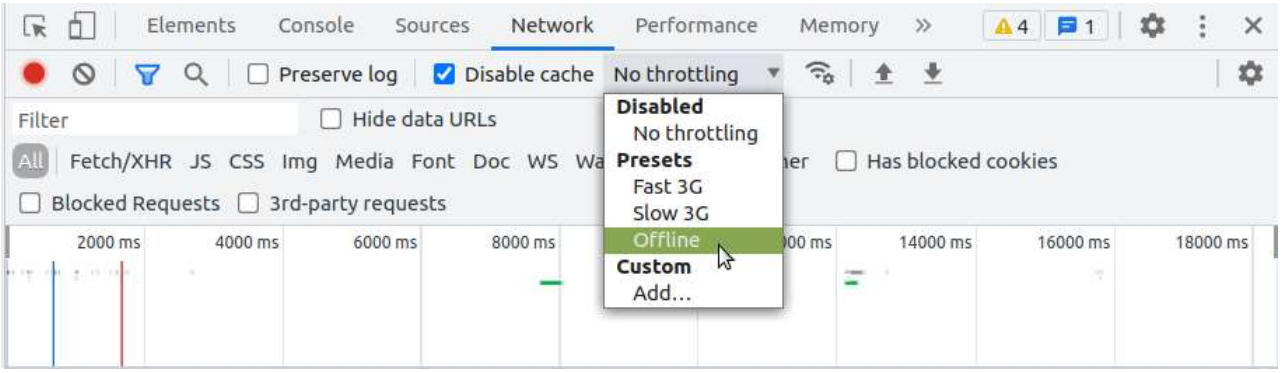
> **Consejo**: Cuando pruebes los service workers de Angular, es una buena idea usar una ventana de incógnito o privada en tu navegador para asegurarte de que el service worker no acabe leyendo de un estado anterior sobrante, lo que puede causar un comportamiento inesperado.
>
> **Nota**: Si no estás usando HTTPS, el service worker sólo se registrará cuando se acceda a la aplicación en localhost.

Simulating a network issue

Para simular un problema de red, desactive la interacción con la red para su aplicación.

En Chrome:

1. Selecciona Tools > Developer Tools (en el menú de Chrome situado en la esquina superior derecha).
2. Ve a la pestaña Network (Red).
3. Seleccione Fuera de línea en el menú desplegable de Aceleración.

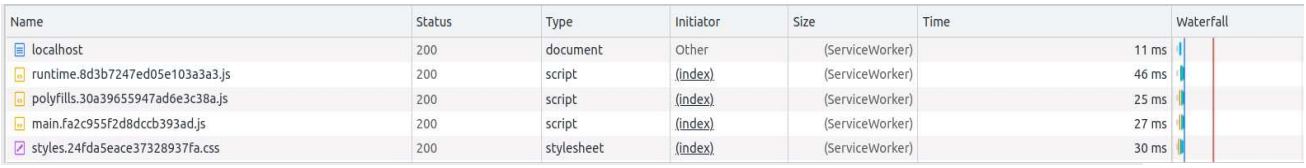


Ahora la aplicación no tiene acceso a la interacción con la red.

Para las aplicaciones que no utilizan el trabajador de servicios de Angular, la actualización ahora mostraría la página de desconexión de Internet de Chrome que dice "No hay conexión a Internet".

Con la adición de un service worker de Angular, el comportamiento de la aplicación cambia. Al actualizar, la página se carga normalmente.

Mire la pestaña Red para verificar que el service worker está activo.



Name	Status	Type	Initiator	Size	Time	Waterfall
localhost	200	document	Other	(ServiceWorker)	11 ms	
runtime.8d3b7247ed05e103a3a3.js	200	script	(index)	(ServiceWorker)	46 ms	
polyfills.30a39655947ad6e3c38a.js	200	script	(index)	(ServiceWorker)	25 ms	
main.fa2c955f2d8dccb393ad.js	200	script	(index)	(ServiceWorker)	27 ms	
styles.24fda5eace37328937fa.css	200	stylesheet	(index)	(ServiceWorker)	30 ms	

Observe que en la columna "Tamaño", el estado de las solicitudes es (ServiceWorker). Esto significa que los recursos no se están cargando desde la red. En su lugar, se están cargando desde la caché del trabajador de servicios.

What's being cached?
Observe que todos los archivos que el navegador necesita para renderizar esta aplicación se almacenan en caché. La configuración ngsw-config.json boilerplate está configurada para almacenar en caché los recursos específicos utilizados por el CLI:

1. index.html.
2. favicon.ico.
3. Artefactos de construcción (paquetes JS y CSS).
4. Todo lo que esté bajo assets.
5. Imágenes y fuentes directamente bajo el outputPath configurado (por defecto ./dist/<nombredelproyecto>/) o resourcesOutputPath.

Préstese atención a dos puntos clave:

1. El ngsw-config.json generado incluye una lista limitada de fuentes y extensiones de imágenes almacenables en caché. En algunos casos, es posible que quieras modificar el patrón glob para adaptarlo a tus necesidades.

2. Si las rutas resourcesOutputPath o assets se modifican después de la generación del archivo de configuración, deberá cambiar las rutas manualmente en ngsw-config.json

Making changes to your application

Ahora que has visto cómo los service workers almacenan en caché tu aplicación, el siguiente paso es entender cómo funcionan las actualizaciones. Realiza un cambio en la aplicación y observa cómo el service worker instala la actualización:

1. Si estás probando en una ventana de incógnito, abre una segunda pestaña en blanco. Esto mantiene el estado de incógnito y de caché vivo durante tu prueba.

2. Cierre la pestaña de la aplicación, pero no la ventana. Esto también debería cerrar las herramientas de desarrollo.
3. Cierre el servidor http.
4. Abra src/app/app.component.html para editarlo.
5. Cambia el texto Welcome to {{title}}! por Bienvenue à {{title}}!.
6. Construya y ejecute el servidor de nuevo

```
ng build
```

```
http-server -p 8080 -c-1 dist/<project-name>
```

Updating your application in the browser

Now look at how the browser and service worker handle the updated application.

1. Abrir http://localhost:8080 de nuevo en la misma ventana. ¿Qué sucede??

¿Qué fue mal? Nada, en realidad. El service worker de Angular está haciendo su trabajo y sirviendo la versión de la aplicación que tiene instalada, aunque haya una actualización disponible. En aras de la velocidad, el service worker no espera a comprobar si hay actualizaciones antes de servir la aplicación que ha almacenado en la caché.
Mira los registros del servidor http para ver que el service worker solicita /ngsw.json. Así es como el service worker comprueba las actualizaciones.

2. Refrescar la pagina.

El service worker instala la versión actualizada de su aplicación en segundo plano, y la próxima vez que se carga o recarga la página, el service worker cambia a la última versión.