2 Primeros pasos para construir un Aplicación web progresiva

En el capítulo 1, aprendió que las aplicaciones web progresivas (PWA) ofrecen un conjunto completamente nuevo de funciones que le permiten crear aplicaciones web rápidas, resistentes y atractivas. En este capítulo, veremos algunas de las mejores prácticas para diseñar su código de interfaz de usuario al crear una PWA. Analizaremos una PWA del mundo real y veremos una descripción general de sus características para obtener información sobre cómo puede crear su propia PWA.

2.1 Construye sobre lo que ya tienes

La cita de Alex Russell en el capítulo 1 (sobre los sitios web que toman sus vitaminas) resume perfectamente las características de una PWA y se relaciona muy bien con cómo me sentí cuando comencé a experimentar con Service Workers. Tan pronto como entendí el concepto básico de cómo funcionaban, se me iluminó la cabeza cuando me di cuenta de lo poderosos que podían ser. A medida que comencé a aprender más y más sobre ellos, comencé a experimentar con cada nueva característica o "vitamina" de PWA a la vez. Aprender cualquier tecnología nueva a menudo puede parecer como escalar una montaña. Pero si aborda su aprendizaje sobre las PWA con la mentalidad de aprender una característica nueva a la vez, dominará el arte de las PWA en poco tiempo.

Sin duda has dedicado mucho tiempo y esfuerzo a tus proyectos actuales. Afortunadamente, crear una PWA no significa que tengas que empezar de nuevo desde cero.
Cuando intento mejorar una aplicación existente, agrego una nueva "vitamina" siempre que siento que beneficiará al usuario y mejorará su experiencia. Me gusta pensar en cada nueva característica de PWA como si Super Mario subiera de nivel cada vez que come un hongo nuevo.

Si tiene una aplicación web existente que cree que se beneficiaría de las funciones de una PWA, le recomiendo que eche un vistazo a una útil herramienta llamada Lighthouse.

(https://github.com/GoogleChrome/lighthouse). Proporciona información útil sobre el rendimiento y la auditoría de su aplicación web, como se muestra en la figura 2.1.

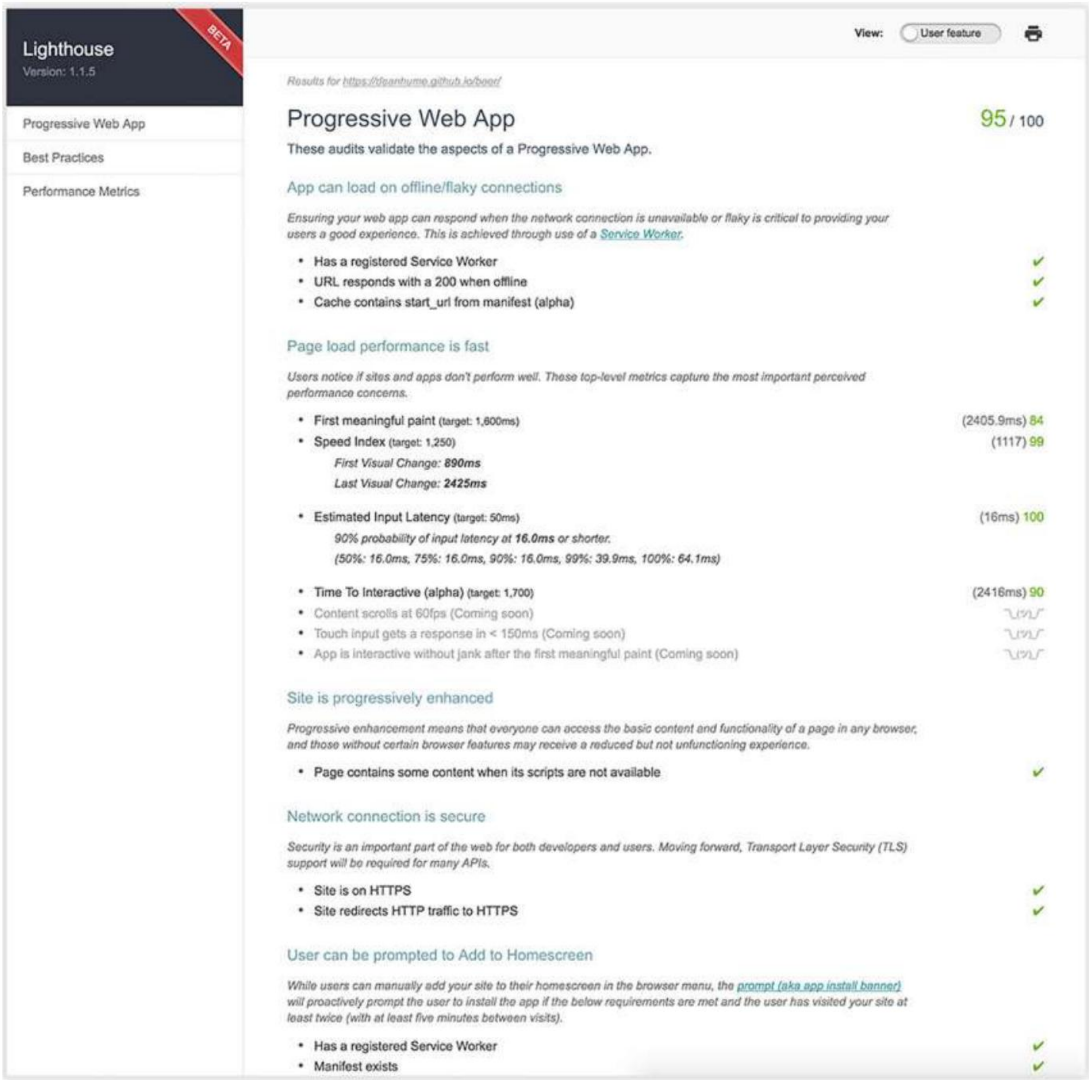

Figura 2.1 La herramienta Lighthouse es excelente para auditar y producir métricas de rendimiento de una aplicación web progresiva.

Puede usarlo como una interfaz de línea de comandos o, si su navegador preferido es Google Chrome, use la práctica extensión de Chrome. Si lo ejecuta mientras apunta a un sitio web, produce algo similar a la figura 2.1. La herramienta realiza una auditoría de su sitio y produce una lista de verificación útil de características y métricas de rendimiento que puede utilizar para mejorar su sitio web. Si desea utilizar esta práctica herramienta y ejecutarla en

uno de sus sitios existentes, diríjase a github.com/GoogleChrome/lighthouse a
Saber más.

Con los comentarios de la herramienta Lighthouse, puede agregar nuevas funciones una por una.
tiempo y mejorar poco a poco la experiencia general de su sitio web.

En este punto, es posible que se pregunte qué función desea agregar a su
sitio existente. Los trabajadores de servicios abren todo un mundo de posibilidades y decidir
por dónde empezar puede ser complicado. A medida que avanzamos en el resto de este libro, cada capítulo
se centrará en una nueva aplicación web progresiva y se escribirá de una manera que le permita
para comenzar a crear con él independientemente de si está creando para un sitio web existente
o uno nuevo.

2.2 Enfoques arquitectónicos iniciales para la construcción de PWA

Entre los desarrolladores, existe un debate en curso sobre si es preferible construir un
aplicación nativa o una aplicación web. Personalmente, creo que debes elegir en función de la
necesidades de sus usuarios. No debería ser un caso de PWA versus aplicaciones nativas, sino más bien, como
Los desarrolladores siempre debemos buscar mejorar la experiencia del usuario. Como puedas
Imagínese, tengo una tendencia natural hacia la creación para la web, pero independientemente de sus preferencias,
si piensa en una PWA como un conjunto de mejores prácticas, creará mejores sitios web. Para
Por ejemplo, si te gusta desarrollar con React o Angular, puedes continuar haciéndolo.
Crear una PWA solo mejorará la aplicación web y la hará más rápida, más atractiva y más resistente.

Los desarrolladores de aplicaciones nativas llevan mucho tiempo ofreciendo a sus usuarios funciones que la web
los desarrolladores sólo podían soñar, como la capacidad de operar sin conexión y responder
independientemente de la conexión de red. Pero gracias a las nuevas características que aportan las PWA
la web, podemos esforzarnos por crear sitios web aún mejores. Muchas aplicaciones nativas tienen una buena
arquitectura y, como desarrolladores web, podemos aprender de sus enfoques arquitectónicos. El
La siguiente sección analiza diferentes enfoques arquitectónicos que puede utilizar en su interfaz.
código cuando se trata de construir PWA.

2.2.1 La arquitectura del shell de la aplicación

Hoy en día hay disponibles muchas aplicaciones nativas excelentes. La aplicación de Facebook, por ejemplo, ofrece
una agradable experiencia para el usuario. Le permite saber cuándo está desconectado, almacena en caché su línea
de tiempo para un acceso más rápido y se carga en un instante. Si no has usado el nativo de Facebook
aplicación dentro de un tiempo, seguirás viendo un shell de interfaz de usuario vacío con el encabezado y la barra de navegación
instantáneamente antes de que se haya cargado el contenido dinámico.

Usando el poder de los Service Workers, no hay razón para que no puedas proporcionar lo mismo
experiencia en la red. Con el almacenamiento en caché inteligente de Service Worker, puede almacenar en caché la interfaz de usuario
caparazón de su sitio web para visitas repetidas. Las nuevas funciones le permiten empezar a pensar en
y construir sus sitios web de manera diferente. Ya sea que esté reescribiendo una aplicación existente o comenzando
desde cero, este enfoque es algo que debe considerar.

Quizás se pregunte qué quiero decir con shell de interfaz de usuario. Me refiero al mínimo HTML, CSS y
Se requiere JavaScript para alimentar la interfaz de usuario. Esto puede ser algo así como el encabezado,

pie de página y navegación de un sitio sin ningún contenido dinámico. Si puede cargar el shell de la interfaz de usuario y almacenarlo en caché, podrá cargar el contenido dinámico en la página más adelante. Un gran ejemplo de esto en acción es la Bandeja de entrada de Google, que se muestra en la figura 2.2.

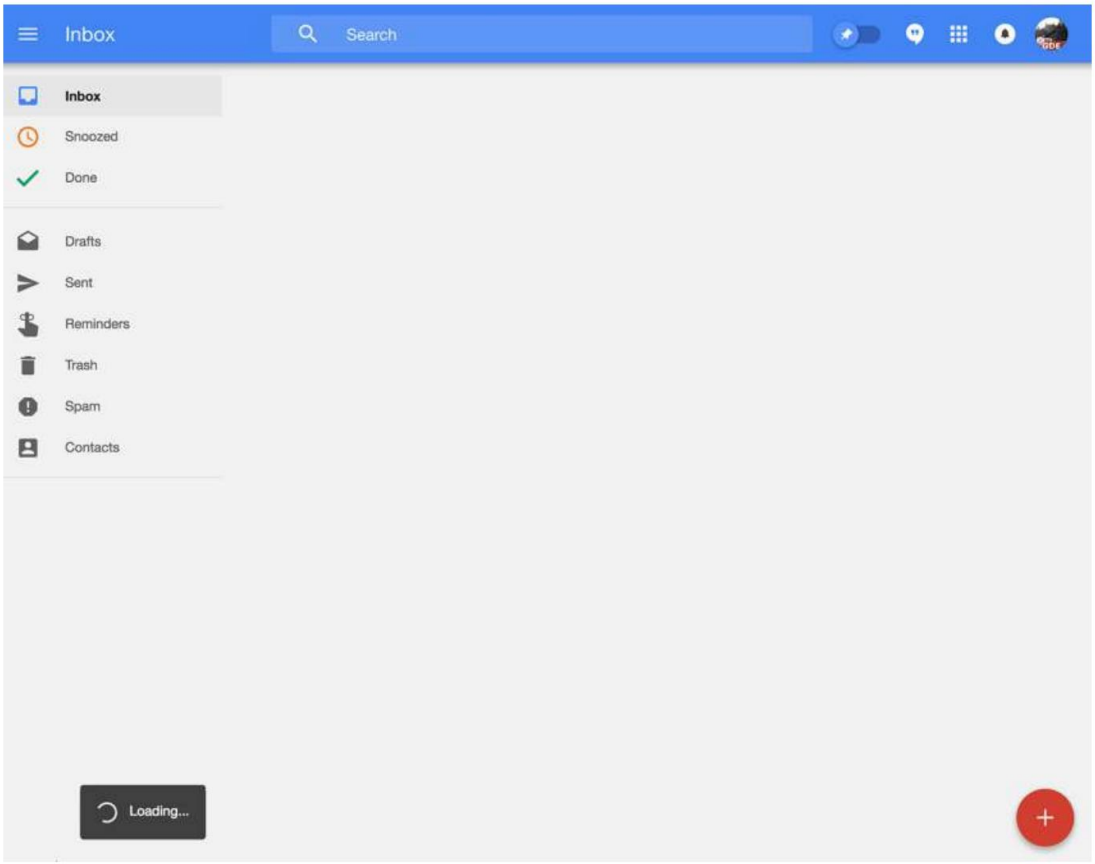

Figura 2.2 La Bandeja de entrada de Google aprovecha los Service Workers para almacenar en caché el shell de la interfaz de usuario.

Es posible que ya esté familiarizado con la Bandeja de entrada de Google, una práctica aplicación web que le permite organizar y enviar correos electrónicos. Debajo del capó, utiliza Service Workers para almacenar en caché y brindar una experiencia súper rápida al usuario. Como puede ver en la figura 2.2, cuando visita el sitio por primera vez, se le presenta instantáneamente la interfaz de usuario del sitio web. Esto es fantástico porque el usuario recibe comentarios instantáneos y el sitio se siente rápido incluso si todavía estás esperando que se cargue el contenido dinámico. La aplicación da la percepción de velocidad incluso si todavía se tarda el mismo tiempo en recuperar el contenido. También se notifica al usuario con un indicador de “cargando” que algo está sucediendo y que el sitio está ocupado; eso es mucho mejor que esperar a que se cargue una página en blanco vacía. Una vez que se ha cargado el shell, los contenidos dinámicos del sitio se obtienen y cargan usando JavaScript, como se muestra en la figura 2.3.

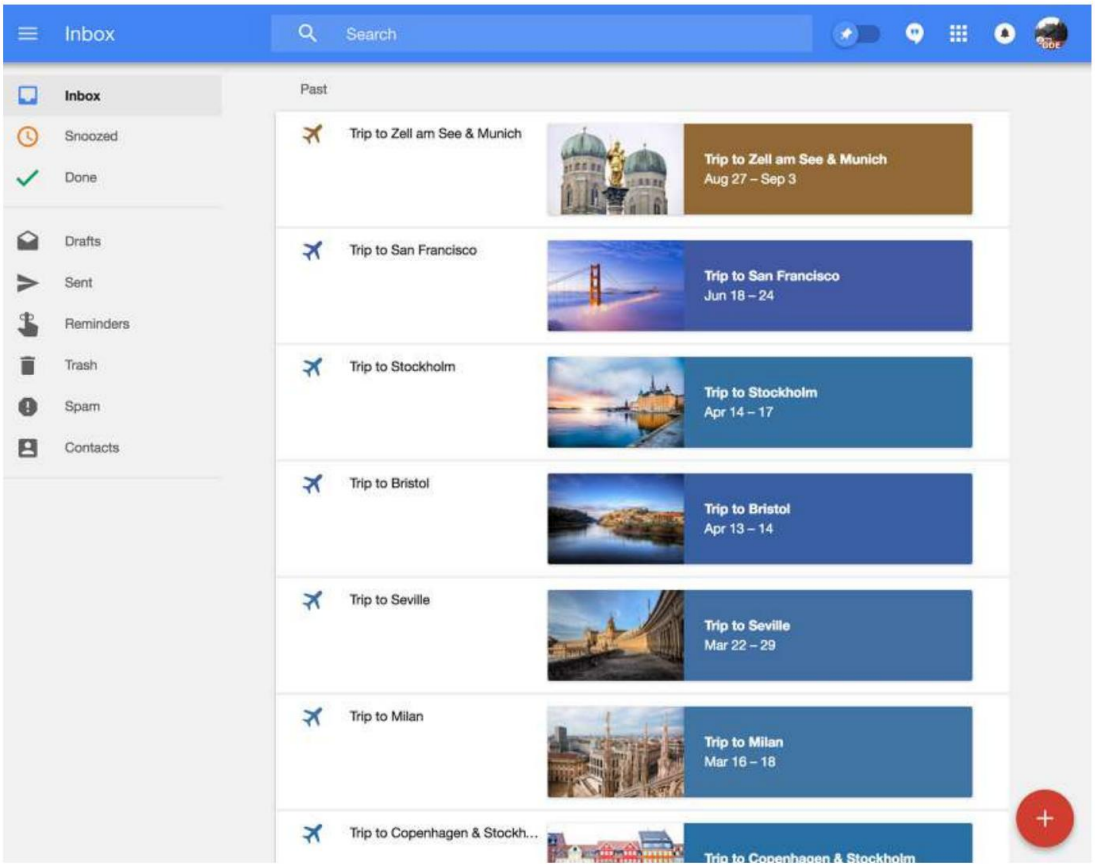

Figura 2.3 Una vez cargado el shell de la interfaz de usuario, el contenido dinámico de un sitio web se recupera y se agrega a la página.

La Figura 2.3 muestra el sitio de la Bandeja de entrada de Google una vez que el contenido dinámico se ha cargado y completado en la aplicación web. Utilizando esta misma técnica, puede proporcionar carga instantánea para visitas repetidas a su sitio web. También puede almacenar en caché el shell de la interfaz de usuario de su aplicación para que funcione sin conexión, lo que le permitirá obtener píxeles significativos en la pantalla incluso si el usuario no tiene una conexión actualmente.

En el capítulo 3, aprenderá cómo aprovechar los Service Workers para almacenar en caché su contenido y brindar una experiencia sin conexión a sus usuarios. A lo largo de este libro, creará una PWA que utilizará la arquitectura de Shell de aplicación. Tendrás la oportunidad de descargar y seguir el código y crear tu propia aplicación utilizando este enfoque.

2.2.2 Beneficios de rendimiento

Es fácil decir que una aplicación web se carga “instantáneamente” usando App Shell Architecture, pero ¿qué significa eso para un usuario? ¿Qué tan rápido es eso? Para ver qué tan rápido usa una PWA

Cuando se carga la arquitectura App Shell, utilicé una herramienta llamada webpagetest.org para producir el tira de película en la figura 2.4, que muestra el tiempo de carga antes y después del almacenamiento en caché usando Ser-vice Workers.

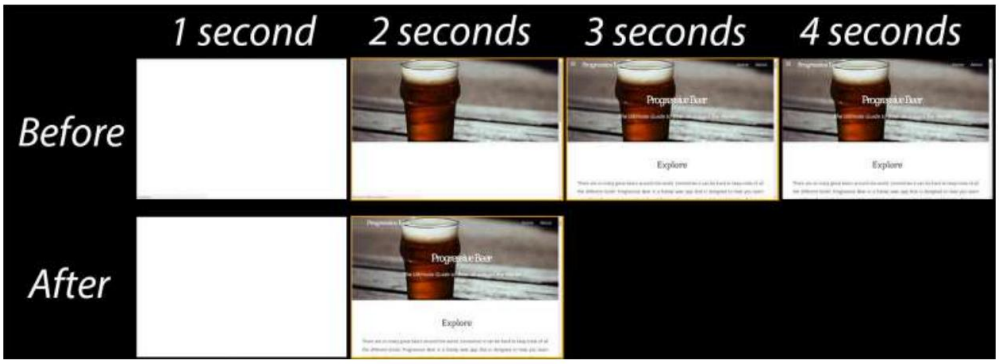

Figura 2.4 La arquitectura de App Shell puede proporcionar al usuario píxeles significativos en la pantalla incluso antes de que el contenido dinámico haya terminado de cargarse.

Ejecuté la herramienta en una PWA que construí llamada Progressive Beer para mostrar una vista de tira de película de un Carga de PWA a lo largo del tiempo. Para el usuario nuevo, el sitio tarda un poco más en descargarse porque está recuperando los recursos por primera vez. Una vez que todos los activos tienen descargado, el usuario nuevo podrá interactuar completamente con el sitio en alrededor de cuatro segundos.

El usuario recurrente que tiene instalado un Service Worker activo ve el shell de la interfaz de usuario en alrededor de 0,5 segundos (500 milisegundos), que se carga a pesar de que la dinámica El contenido aún no ha sido devuelto por el servidor. Luego el resto del El contenido dinámico se carga y se completa en la pantalla. Lo mejor de esto enfoque es que incluso si no hay conexión, un usuario aún puede ver el shell UI del sitio en aproximadamente medio segundo, momento en el cual podrías presentarles algo significativo, como notificarles que están desconectados o proporcionarles contenido almacenado en caché.

Cada vez que vuelvan a visitar el sitio, tendrán esta experiencia mejorada que es rápida, confiable y atractivo. Si se está acercando al desarrollo de una nueva aplicación web, El uso de la arquitectura Application Shell puede ser una forma eficiente de aprovechar Trabajadores de servicios.

2.2.3 La arquitectura del shell de aplicaciones en acción

En el capítulo 1, repasamos las distintas etapas del ciclo de vida del Service Worker. Puede No habría tenido mucho sentido en ese momento, pero a medida que profundizamos un poco más en cómo La arquitectura Shell de la aplicación funciona, lo hará. Recuerde que al utilizar un trabajador de servicio, puede aprovechar los diferentes eventos en el ciclo de vida del trabajador del servicio. La figura 2.5 ilustra cómo aprovechar estos eventos.

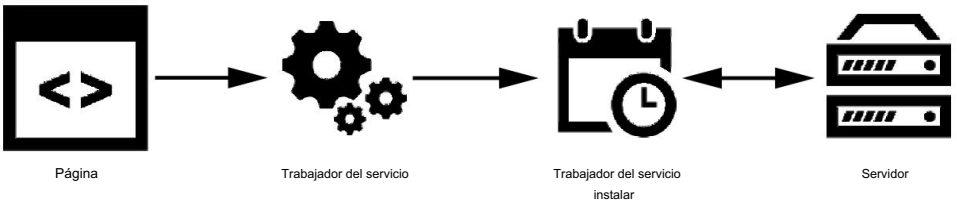

Figura 2.5 Durante el paso de instalación de Service Worker, puede recuperar recursos y preparar el caché para la próxima visita.

Cuando el usuario visita el sitio web por primera vez, el Service Worker comienza a descargarse e instalarse. Durante la etapa de instalación, puedes acceder a este evento.
y preparar el caché con todos los recursos necesarios para el shell de la interfaz de usuario: el HTML básico página y cualquier CSS o JavaScript que pueda ser necesario.

Ahora puede servir el "shell" del sitio instantáneamente porque se agregó al Caché del trabajador de servicio. La solicitud HTTP para estos recursos nunca necesita ir al servidor nuevamente. Tan pronto como el usuario navega a otra página, verá el shell sin demora (ver figura 2.6).

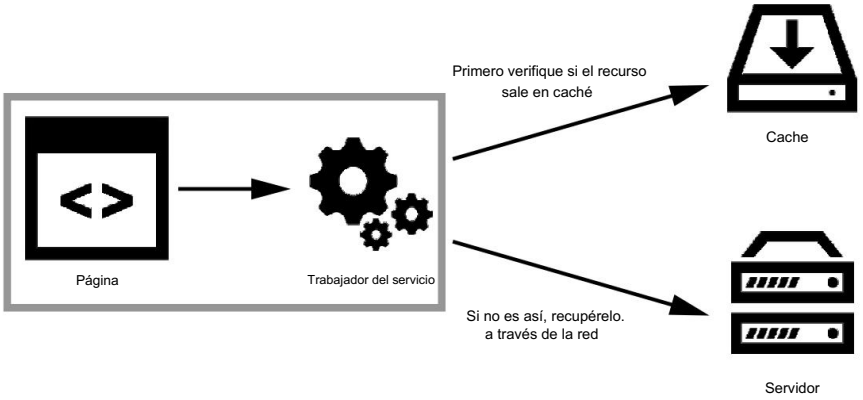

Figura 2.6 Para cualquier solicitud HTTP que se realice, puede verificar si el recurso ya existe en la caché y, si no es así, recuperarlo a través de la red.

El contenido dinámico que se cargará en el sitio podrá continuar con normalidad. Como accede al evento de recuperación para estas solicitudes, puede decidir en este punto si desea almacenarlos en caché o no. Es posible que tengas contenido dinámico que se actualiza con frecuencia, por lo que puede que no tenga sentido almacenarlo en caché. Pero sus usuarios seguirán recibiendo una información más rápida y mejorada. experiencia de navegación. En el capítulo 3, profundizaremos en el almacenamiento en caché de Service Worker.

2.3 Diseccionar una PWA existente paso a paso

Aunque es un concepto relativamente nuevo, algunas PWA sorprendentes ya están en la web. siendo utilizado por millones de usuarios cada día.

En el capítulo 3, comenzarás a construir tu propia PWA. Primero, analicemos una PWA existente.
para ver cómo funcionan algunas de estas características.

En esta sección, veremos una de mis PWA favoritas, el sitio web móvil de Twitter, una PWA que ofrece a los usuarios una experiencia mejorada en sus dispositivos móviles. Si usa Twitter, es una excelente manera de ver sus tweets mientras viaja (consulte la figura 2.7).

Si navegas a twitter.com en su dispositivo móvil, será redirigido a mobile.twitter.com y se muestra un sitio diferente. Twitter llamó a su PWA Twitter Lite porque ocupa menos de un megabyte de almacenamiento y afirma que puede ahorrar hasta un 70% en datos mientras se carga un 30% más rápido.

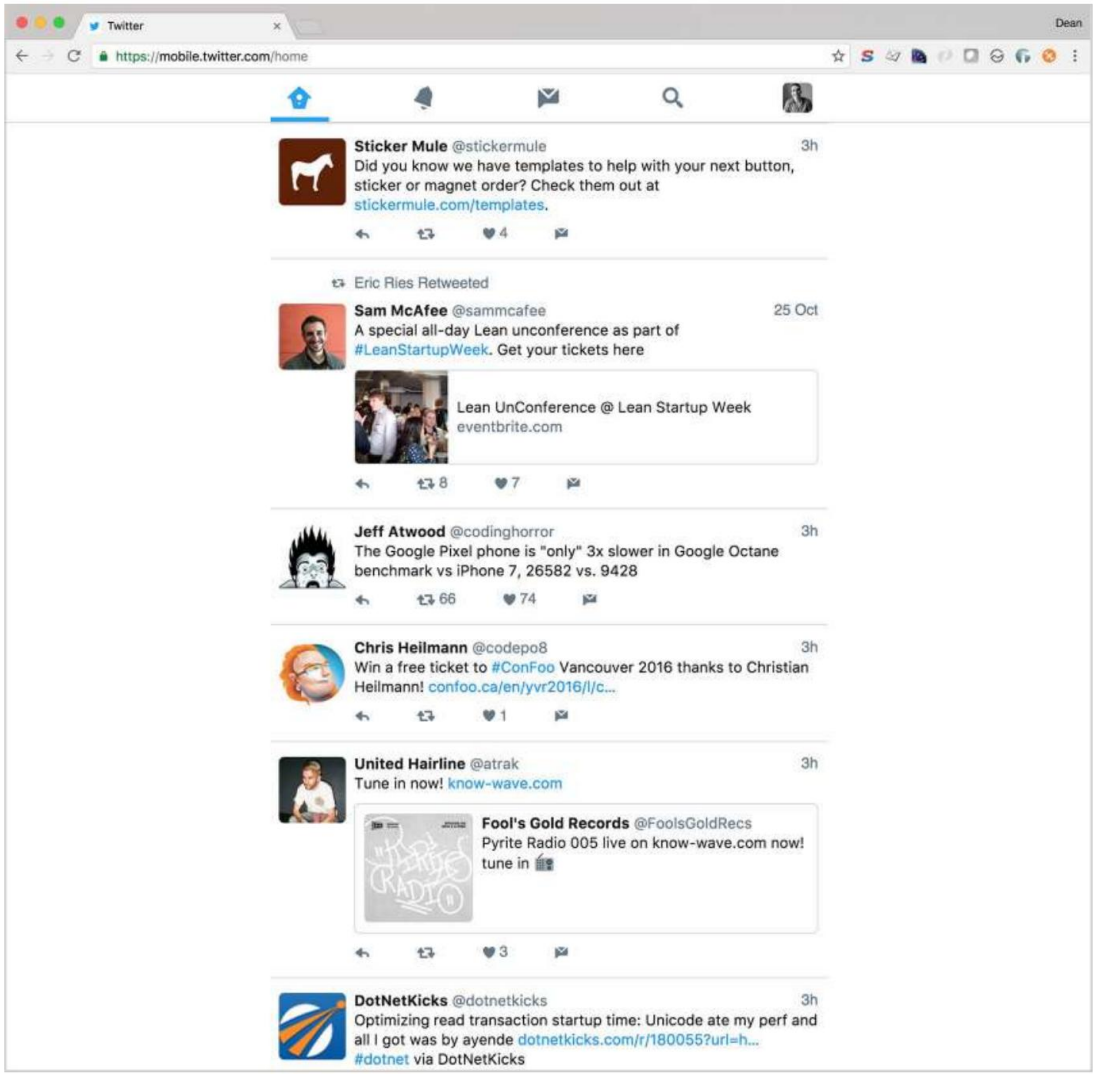

Figura 2.7 El sitio web móvil de Twitter es una PWA que aprovecha la arquitectura Application Shell.

Personalmente, creo que es tan bueno que debería usarse también en la web de escritorio. Prefiero usar la versión PWA a la versión nativa; puedes acceder a la aplicación web en tu dispositivo de escritorio navegando directamente a mobile.twitter.com.

2.3.1 Arquitectura de interfaz de usuario

En el fondo, Twitter Lite está construido utilizando una arquitectura de Shell de aplicación. Esto significa que utiliza una página HTML simple para la interfaz de usuario del sitio y el contenido principal de la página se inyecta dinámicamente mediante JavaScript. Si el navegador del usuario admite Service Workers, todos los recursos necesarios para el shell de la interfaz de usuario se almacenan en caché durante la instalación de Service Worker.

Para los visitantes habituales, esto significa que el shell se carga en un instante (ver figura 2.8). Este enfoque seguirá funcionando en navegadores que no admitan Service Workers; no tendrán los activos para el shell de la interfaz de usuario almacenados en caché y perderán la ventaja adicional de rendimiento súper rápido. La aplicación web también se ha optimizado para una variedad de tamaños de pantalla diferentes mediante un diseño web responsivo.

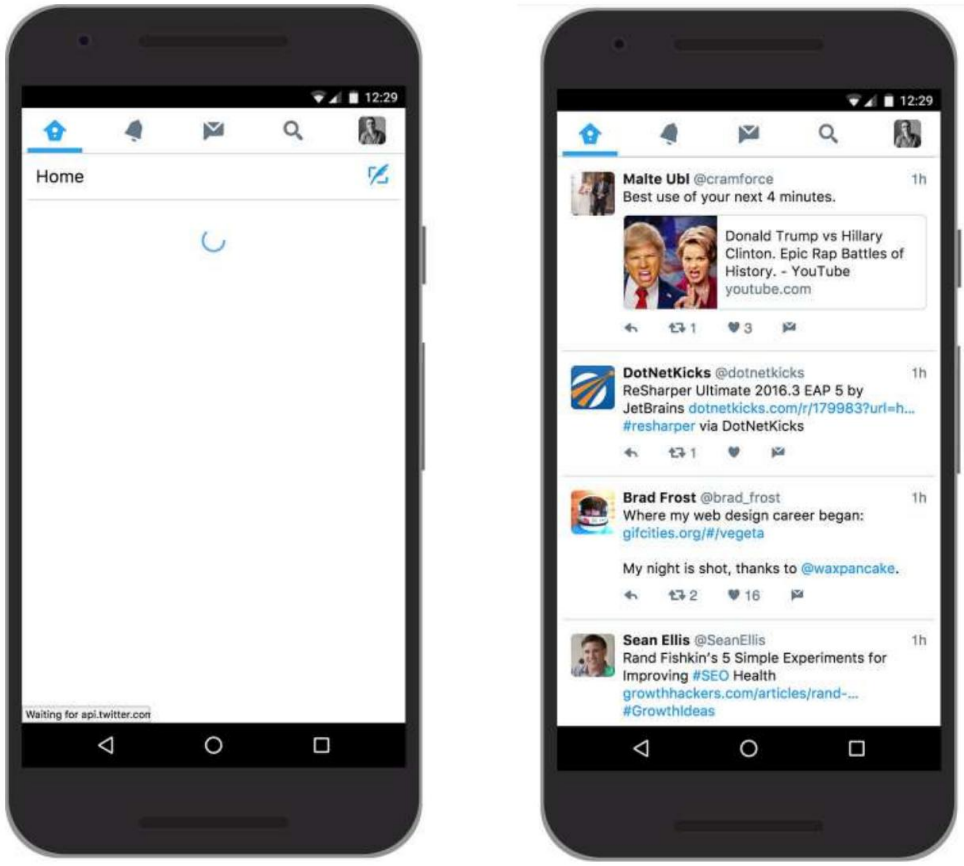

Figura 2.8 La arquitectura App Shell aporta instantáneamente píxeles significativos a la pantalla. La imagen de la izquierda es lo que el usuario ve primero y luego ve la pantalla de la derecha una vez cargada.

2.3.2 Almacenamiento en caché

Service Worker Caching es una característica poderosa que brinda a los desarrolladores web la capacidad de almacene en caché mediante programación los recursos que necesita. Puede interceptar solicitudes HTTP y respuestas y ajústelas como mejor le parezca, y esta es la clave para desbloquear aún mejores aplicaciones web. El uso de un trabajador de servicio le permite acceder a cualquier solicitud de red y decide exactamente cómo quieres responder. Crear una PWA rápida y resistente es fácil utilizando el almacenamiento en caché de Service Worker.

Twitter Lite es rápido. Se siente bien de usar y las páginas que se han almacenado en caché se cargan. casi al instante. Como usuario, es el tipo de experiencia que me gustaría esperar de cada sitio web.

Al momento de escribir esto, Twitter Lite utiliza una útil biblioteca llamada Servicio Worker Toolbox que contiene técnicas de almacenamiento en caché probadas y comprobadas que utilizan el Servicio Trabajadores. Esta caja de herramientas le proporciona algunas ayudas básicas para que pueda empezar a crear sus propios trabajadores de servicio y le evita escribir código repetitivo. En el capítulo 3, Profundizaremos en el almacenamiento en caché y, sin adelantarnos demasiado, echemos un vistazo a Ejemplo de almacenamiento en caché utilizando Service Worker Toolbox. La aplicación Twitter PWA está usando esto técnica para almacenar en caché sus emojis. No te preocupes si el código en el siguiente listado no cumple sentido en este momento; profundizaremos en esto en el capítulo 3.

Listado 2.1 El código del trabajador del servicio Twitter Lite

```
toolbox.router.get("/emoji/v2/svg/:icono", función(evento) {
    return caches.open('twemoji').luego(función(respuesta) {
        devolver respuesta.partido(evento.solicitud).luego(función(respuesta) {
            respuesta de devolución || buscar (evento.solicitud)
        })
    }).catch(función() {
        recuperación de retorno (evento.solicitud)
    })
}, {
    origen: /abs.*\.twimg\.com$/
})
```

Interceptar cualquier solicitud de ruta /emoji/v2/svg/:icono.

Abra un caché existente llamado twemoji.

Compruebe si el solicitud actual coincide con cualquier cosa en tu caché.

Si es así, devuélvalo inmediatamente. De lo contrario, continúe con normalidad.

Si algo sale mal al abrir el caché, continúa normalmente.

También solo quieres comprobar por este recurso contra el dominio twimg.com.

En el listado 2.1, Service Worker Toolbox busca cualquier solicitud entrante que coincide con la URL /emoji/v2/svg/ y proviene de un origen de *.twimg.com. una vez que intercepta cualquier solicitud HTTP que coincida con esta ruta, la almacenará en caché con el nombre wemoji. La próxima vez que un usuario realice una solicitud para la misma ruta, se le presentará el resultado almacenado en caché.

Este poderoso fragmento de código le brinda a usted, como desarrollador, la capacidad de controlar exactamente cómo y cuándo desea almacenar en caché los activos en su sitio. No te preocupes si ese código

Parece un poco confuso al principio. En el capítulo 3, creará una página que utilice esta poderosa característica.

2.3.3 Navegación sin conexión

En mi viaje diario hacia y desde el trabajo, Tomo el tren. Tengo suerte de que el viaje no sea demasiado largo, pero lamentablemente el La señal de la red es débil en algunas áreas y puede caer. Esto significa que si estoy navegando por la web en mi teléfono, puedo perder la conexión o la conexión puede volverse escamoso. Puede resultar bastante frustrante.

Afortunadamente, el almacenamiento en caché de Service Worker guarda activos en el dispositivo del usuario. Usando Trabajadores de servicio, puedes interceptar cualquier Solicitudes HTTP y respuesta directa desde el dispositivo. Ni siquiera necesitas acceso a la red para recuperar datos almacenados en caché activos.

Con esto en mente, puedes construir páginas sin conexión. Usando el almacenamiento en caché de Service Worker, puede almacenar en caché recursos individuales o incluso páginas web enteras, tú decides. Por ejemplo, si el usuario no tiene conexión, Twitter Lite presenta una página sin conexión personalizada. página, como se muestra en la figura 2.9.

En lugar de ver el temido "Esto Error "no se puede acceder al sitio", el usuario ahora ve una útil página personalizada sin conexión o una versión en caché de una página que ya han visitado. El usuario puede comprobar si La conectividad se ha restablecido tocando el botón proporcionado. Para el usuario, esto es una mejor experiencia web. En el capítulo 8, dominarás las habilidades necesarias requeridas para comenzar a crear tus propias páginas sin conexión y proporcione a sus usuarios una solución resistente experiencia de navegación.

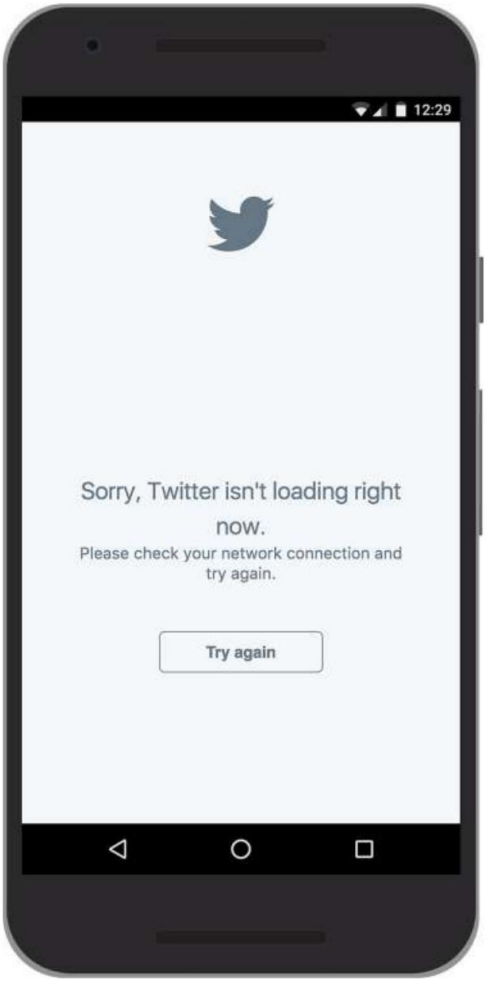

Figura 2.9 Si un usuario no tiene conexión, Twitter PWA muestra una página de error personalizada.

2.3.4 Aspecto y sensación

Twitter Lite es rápido, está optimizado para pantallas más pequeñas y funciona sin conexión. ¿Qué más queda? Bueno, debe verse y sentirse como una aplicación. Si observa detenidamente el HTML de la página de inicio de la aplicación web, es posible que observe la siguiente línea:

```
<enlace rel="manifest" href="/manifest.json">
```

Este enlace apunta a un archivo conocido como archivo de manifiesto. Este es un archivo JSON simple que sigue la especificación del Manifiesto de aplicación web del W3C1 y brinda al desarrollador control sobre diferentes elementos de la apariencia de la aplicación. Proporciona información como nombre, autor, icono y descripción de su aplicación web. También le brinda algunos otros beneficios. En primer lugar, permite al navegador instalar la aplicación web en la pantalla de inicio de un dispositivo para brindar a los usuarios un acceso más rápido y una experiencia más rica. Además, al establecer un color de marca en el archivo de manifiesto, puede personalizar la pantalla de presentación que presenta automáticamente el navegador. Y le permite personalizar la barra de direcciones de su navegador para que coincida con los colores de su marca.

El uso de un archivo de manifiesto completa la apariencia de su aplicación web y proporciona a los usuarios una experiencia más rica. Twitter Lite utiliza un archivo de manifiesto para aprovechar muchas de las funciones que se integran automáticamente en el navegador.

En el capítulo 6, exploraremos cómo puede utilizar el archivo de manifiesto para mejorar la apariencia de su PWA y brindar a sus usuarios una experiencia de navegación atractiva.

2.3.5 El producto final

Twitter Lite es un ejemplo completo de lo que debería ser una PWA. Cubre la mayoría de las funciones que analizaremos en este libro para crear una aplicación web rápida, atractiva y confiable.

En el capítulo 1, hablamos de todas las características de lo que debería ser una aplicación web. Repasemos el desglose de la PWA de Twitter hasta ahora. La aplicación se puede describir de la siguiente manera:

Responsive: se ajusta a tamaños de pantalla más pequeños.
Independiente de la conectividad: funciona sin conexión debido al almacenamiento en caché de Service Worker. Interacciones similares a las de una aplicación: se crea utilizando la arquitectura de Shell de aplicación. Siempre actualizado: se actualiza gracias al proceso de actualización de Service Worker. Seguro: funciona a través de HTTPS. Descubrible:
un motor de búsqueda puede encontrarlo. Instalable: se
puede instalar mediante el archivo de manifiesto. Vinculable: se puede
compartir fácilmente mediante URL.

¡Vaya! Esa es una lista larga: muchas de estas cosas las obtenemos como efectos secundarios de la creación de una PWA.

---

[1] www.w3.org/TR/appmanifest/

2.4 Resumen

Las funciones que las aplicaciones web progresivas aportan a la web le permiten a usted, como desarrollador, Cree mejores sitios web que sean más rápidos, confiables y atractivos para sus usuarios.

Estas funciones están integradas en el navegador, lo que significa que también funcionan muy bien. con cualquier biblioteca o marco con el que esté familiarizado. Independientemente de si tiene una aplicación existente o le gustaría crear una nueva aplicación web desde cero, puede Adapte una PWA a sus necesidades.

Este capítulo analizó la arquitectura de Shell de aplicaciones, un enfoque que puede utilizar aprovechar el almacenamiento en caché de Service Worker para proporcionar a sus usuarios información significativa píxeles al instante.

Analizamos la PWA de Twitter y analizamos muchas de las funciones que están disponibles. en su navegador ahora mismo.

A lo largo del resto de este libro, profundizaremos en cada una de estas características una por una, y aprenderá cómo crear una PWA sencilla y eficaz como Twitter Lite.