8VO CONGRESO NACIONAL

INGENIERÍA INFORMÁTICA / SISTEMAS DE INFORMACIÓN

VIRTUAL CONAIISI 2020

50 UTN ANIVERSARIO FACULTAD REGIONAL SAN FRANCISCO

www.sanfrancisco.utn.edu.ar

05 06 NOV.

RIISIC

confedi Consejo Federal de Decanos de Ingeniería República Argentina

150 ING la Argentina celebra su ingeniería 1870-2020

Octavo Congreso Nacional de Ingeniería Informática/Sistemas de información

CONAIISI

5 y 6 de noviembre de 2020

Universidad Tecnológica Nacional, Facultad Regional San Francisco

Memoria de Trabajos

Claudia Verino, Juan Carlos Calloni, Gabriel Cerutti, Alfonsina E. Andreatta
(Compiladores)

San Francisco, Córdoba - Argentina, Marzo de 2021

Octavo Congreso Nacional de Ingeniería Informática/Sistemas de Información: CONAIISI.
5 y 6 de noviembre de 2020 / Claudia Verino, Juan Carlos Calloni, Gabriel Cerutti, Alfonsina E. Andreatta; compilado por Claudia Verino , Juan Carlos Calloni, Gabriel Cerutti, Alfonsina E. Andreatta. - 1a ed. -
Ciudad Autónoma de Buenos Aires: Universidad Tecnológica Nacional.
Facultad Regional San Francisco, 2021.
Libro digital, PDF

Archivo Digital: descarga
ISBN 978-950-42-0202-8

1. Sistemas de Información. 2. Ingeniería Informática. I. Verino, Caludia, Calloni, Juan Carlos, Cerutti, Gabriel, Andreatta, Alfonsina E. comp.
CDD 004.07

Octavo Congreso Nacional de Ingeniería Informática/Sistemas de información
CONAIISI
5 y 6 de noviembre de 2020
Universidad Tecnológica Nacional, Facultad Regional San Francisco

Memorias de trabajo

Diseño de Tapa: María Laura Vaudagna

ISBN 978-950-42-0202-8

Auspiciantes:

Estrategias de Manejo de Cache para Aplicaciones Web Progresivas - Presentación de un Esquema optimizado

Pablo Martín Vera, Rocío Rodríguez
Universidad Abierta Interamericana
Centro de Altos Estudios en Tecnología Informática (CAETI)
Av. Montes de Oca 745, Ciudad Autónoma de Buenos Aires, Argentina
pablomartin.vera@uai.edu.ar; rocioandrea.rodriguez@uai.edu.ar

Resumen

Las aplicaciones web progresivas permiten tomar las ventajas del diseño adaptativo construyendo una única solución que pueda ser utilizada en todas las plataformas, pero además tiene características propias de las aplicaciones nativas. Algunas de estas características son iniciarlas desde un ícono, trabajar desconectadas, manejar almacenamiento local en el dispositivo, así como acceso al hardware. En este artículo se presenta una estrategia para el manejo de cache en aplicaciones web progresivas, que permite evitar descargar datos de la red cuando estos pueden ser accedidos desde la cache del dispositivo aun cuando se detecte una nueva versión.

Palabras clave: Aplicaciones Web Progresivas, Cache, Dispositivos Móviles,

1. Introducción

Dada la amplia inserción de dispositivos móviles estos han sido foco necesario de análisis al momento de diseñar y desarrollar aplicaciones. Cada vez son menos los usuarios que tienen una computadora de escritorio o notebook a disposición y en contrapartida aumenta el número de usuarios que realizan un sin fin de tareas desde sus teléfonos celulares. Un estudio estadístico a nivel mundial [1] indica que el porcentaje de personas entre 16 y 64 años que tienen un teléfono móvil es del 76%, notebook 45%, computadora de escritorio 32%. "La rápida expansión y adopción de los teléfonos móviles han generado cambios sociales y culturales en la sociedad, han modificado las formas de comunicación, de acceder a la información y las maneras en que los individuos se relacionan entre sí. La telefonía móvil ha producido una transformación en los ritos sociales de interacción. El uso de las TIC ha modificado la forma de trabajar, aprender, colaborar, jugar, pasar el tiempo y socializar de los individuos" [2].

El desarrollo de aplicaciones nativas se vio condicionado con el esfuerzo que conlleva realizar aplicaciones distintas para diferentes sistemas operativos, o incluso versionados de un mismo sistema operativo. "Las aplicaciones móviles han cambiado el mercado tecnológico […] accesibilidad para cualquier usuario, nuevos trabajos y contenido siempre accesible en cualquier momento [...] Este hecho provoca que las empresas se vean obligadas a tener sus aplicaciones disponibles en línea para cualquier dispositivo independientemente de su sistema operativo, el tamaño de la pantalla o cualquier otra condición del dispositivo. Esta imposición (del mercado) naturalmente implica desarrollos, muchos extremadamente caros, en caso de que tenga que desarrollar el producto de forma nativa para cada plataforma" [3]. Por ello los frameworks híbridos permitieron subsanar los tiempos que conllevan los desarrollos y ofrecer con un esfuerzo reducido obtener versiones implementables sobre distintos sistemas operativos. Actualmente el mercado se encuentra dividido mayormente entre Android y iOS, si bien Android lidera el mercado cabe destacar que este sistema operativo se encuentra en dispositivos de diversas marcadas, es por ello por lo que iOS siendo exclusivo de Apple lidera el resto del mercado. A partir de los datos publicados por la empresa de estadísticas statcounter [4] en el mundo el 74,25% corresponde a Android y el 25,15% a iOS. Considerando los datos publicados [4] se realizó la tabla 1 en donde se presentan los porcentajes por regiones.

Tabla 1. Sistemas Operativos, porcentaje por región

Región	Android	iOS	Otros
África	86,89	10,57	2,54
América del Norte	46,98	52,79	0,23
América del Sur	88,52	11,18	0,30
Asia	82,80	16,56	0,64
Europa	73,18	26,39	0,43
Oceanía	49,39	50,19	0,23

Como puede observarse en la tabla 1 se encuentran en gris las celdas que se corresponden con el mayor porcentaje, en los casos en que iOS es mayoritario pude verse que supera el 50% sin llegar al 60% siendo mucho mayor la diferencia en las regiones donde es mayoritario Android superando en todos los casos el 73%. Cabe destacar que en

América del Sur Android tiene aproximadamente un 89% de inserción.

No sólo los usuarios tienen un determinado sistema operativo sino también una versión particular. Pero una organización que quiere tener una aplicación móvil ya no debería pensar en sistemas operativos o versiones, tampoco decidirse por una aplicación instalable o una aplicación web. Las PWA (Aplicación Web Progresiva) permiten ofrecer en un único desarrollo una aplicación que se adapta a todos los dispositivos. “Esta nueva la tecnología permite que una aplicación esté disponible en cualquier dispositivo con acceso a un navegador web, sin la necesidad de desarrollar la aplicación de forma nativa específicamente para un dispositivo o sistema operativo específico” [5].

Este artículo se encuentra estructurado de la siguiente manera, en la sección 2 se presenta un estado del arte mencionando trabajos específicos sobre PWA, en la sección 3 se aborda la temática de PWA mostrándose a través de unos ejemplos por medio de capturas como es la adaptación que puede efectuarse sobre una solución web; en la sección 4 se listan los principios, características y consideraciones técnicas que distinguen a una PWA de una aplicación web, en la sección 5 se abordan los esquemas de cache existentes, en la sección 6 se explica el funcionamiento del service worker que es un proceso que se ejecutará en segundo plano y permitirá llevar a cabo el esquema de cache previsto; en la sección 7 se presenta una propuesta realizada en el marco del proyecto de investigación que detalla como recurrir a una solución que combine estrategias de cache existentes; finalmente en la sección 8 se presentan las conclusiones y trabajos futuros.

2. Estado del Arte

Diversos trabajos académicos han puesto su foco en las Aplicaciones Web Progresivas, como alternativa a las aplicaciones nativas móviles dado que la brecha que las distanciaba se fue achicando y actualmente la web permite tener la apariencia y ventajas de una aplicación nativa, entre estos trabajos se encuentra [5].

Otros autores exponen soluciones puntuales desarrolladas con PWA para algún campo específico, entre ellos: Comercio electrónico [6], Docencia [7], Agricultura [8], Pesca [9], esto permite evidenciar la adopción de PWA en todas las áreas, por otra parte el artículo [10] expone la importancia de las PWA en aplicaciones dedicadas al área de la salud.

Pero es necesario, además de construir aplicaciones PWA entender su funcionamiento y poder mejorarlas, nuestro objetivo está puesto en los esquemas de cache y proponer una estrategia para su optimización. En el artículo [11] se realiza mediciones sobre el acceso y uso de cache comparando aplicaciones nativas con su contrapartida web construida mediante una PWA. El análisis que realiza les permite concluir que la performance de la PWA es superior a la versión nativa y que el uso de cache incrementa notablemente dicha performance. Justamente en el uso de la cache está puesto el foco de este artículo, presentar una estrategia que permita optimizar el uso de la cache para reducir el consumo de datos en cada acceso a una PWA e incluso ante el cambio de versionado.

3. Aplicaciones Web Progresivas

Las PWA se apoyan sobre los principios del diseño web adaptativo. “El diseño web adaptativo o adaptable (en inglés, Responsive Web Design) es una técnica de diseño y desarrollo web que, mediante el uso de estructuras e imágenes fluidas, así como de media-queries en la hoja de estilo CSS, consigue adaptar el sitio web al entorno del usuario” [12].

En la figura 1 se muestra un sitio web diseñado para una PC de escritorio al visualizar el sitio en una pantalla pequeña no se aprovecha el tamaño de la pantalla.

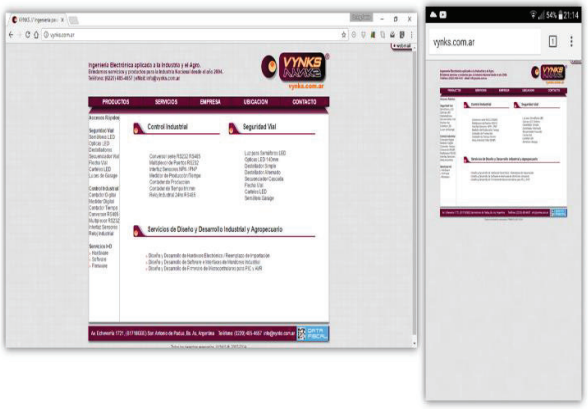

Figura 1. Ejemplo de página web con un tamaño fijo

Para los sitios web desarrollados en HTML 5 tan sólo una línea de código permite ajustar la página para que ocupe todo el ancho del dispositivo y no deje los márgenes que pueden observarse en el ejemplo de la figura 1, dicha línea se muestra en la figura 2.

```
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
```

Figura 2. Línea para ajustar al ancho de Pantalla

Luego se pueden crear los CSS necesarios para aplicar distintos estilos según el tamaño de pantalla de cada dispositivo. “Básicamente, se crean puntos de ruptura o breakpoints en las hojas de estilos CSS, es decir, detectan el tipo de dispositivo y en base a ello el contenido consigue adaptarse a un determinado tamaño de pantalla. Algunas consideraciones pueden ser el ancho y alto del browser, o bien del ancho y alto del dispositivo móvil. Estos son los puntos donde el diseño cambiará de forma, es decir, se adaptará a los distintos anchos de pantallas y resoluciones” [13].

En contrapartida al sitio presentado en la figura 1, en la figura 3 se muestra la versión de escritorio y como se adapta ese sitio a una versión de tamaño menor. El menú de opciones que se visualiza en una computadora se transforma en una barra de accesos rápidos por medio de íconos más un botón que permite acceder al resto de opciones del menú (ver sectores identificados con el número 1). Las opciones que se presentan identificadas con el número 2 en la primera imagen se han encolumnado en la segunda. Todas estas adaptaciones favorecen la visualización en las distintas pantallas.

Figura 3. Ejemplo de Diseño Web Adaptativo

En su artículo «How we built Twitter Lite», Nicolas Gallagher, un desarrollador de dicha red social, explica cómo gracias a la versión PWA de la aplicación consiguieron reducir el consumo de datos hasta fracciones irrisorias en comparación con sus aplicaciones nativas [14], [15].

En la figura 4, se muestra a modo de ejemplo una PWA al ingresar por primera vez a la web aparece debajo un mensaje con el ícono de la aplicación preguntando al usuario si desea agregarla a la pantalla principal.

Una vez instalada se puede acceder desde un ícono a la misma y ya se no se muestra la barra del navegador (se ha elegido ocultarla) y en el caso de la pantalla de la derecha de la figura 5 se ha integrado la barra de título con la de estado con los íconos clásicos del sistema operativo. La forma final que se visualizará será configurada por los desarrolladores según un archivo de manifiesto, donde entre otras cosas se puede seleccionar: modo de visualización, orientación de la pantalla, íconos, color de fondo, color de tema, idioma, nombre corto para la aplicación…

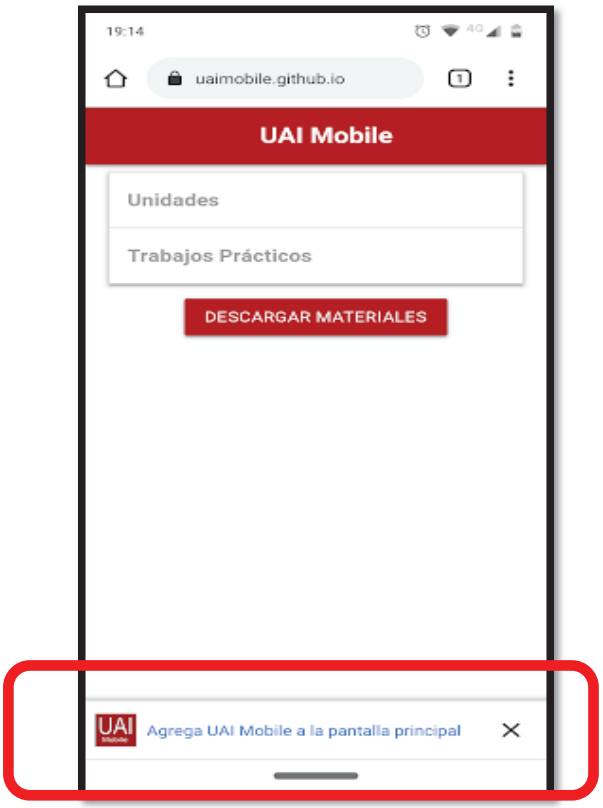

Figura 4. Ejemplo PWA - Instalación

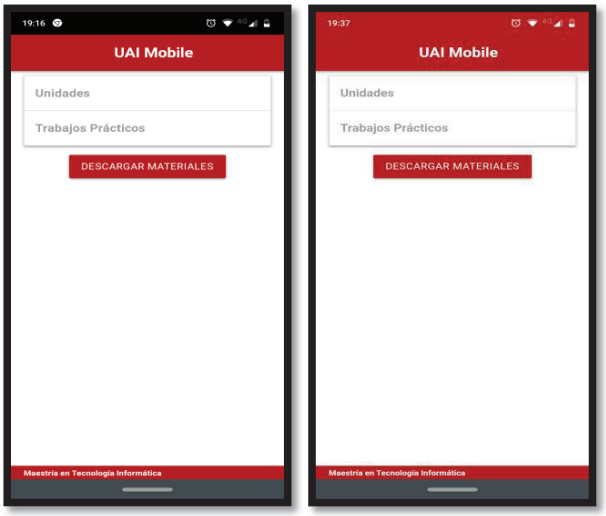

Figura 5. Ejemplo PWA - Visualización

4. Principios de una PWA

Las PWA son aplicaciones web que utilizan las últimas tecnologías para brindar una mejor experiencia de usuario y deben tener las siguientes características:

- Aspecto similar a las apps nativas
- Cargar inmediatamente, independientemente del estado de la red
- Brindar una experiencia de uso inmersiva al ejecutarse en pantalla completa y sin la barra del navegador
- Permitir trabajar sin conexión
- Mostrar notificaciones aun cuando el navegador está cerrado
- Configurar el ícono con que se instalará en el dispositivo

"La PWA se basa en los conceptos de una sola aplicación para todas las plataformas al igual que el enfoque híbrido. Sin embargo, posee distintas capacidades, como carga instantánea, notificaciones push incluso en estado fuera de línea" [16]. También es posible desde las PWA acceder al hardware del dispositivo (sensores, cámara, almacenamiento interno), lo cual resulta imprescindible para muchas aplicaciones [17].

Para que una aplicación pueda tener las características previamente mencionadas es preciso tomar en cuenta diversas pautas al momento de su desarrollo. Resulta interesante un checklist provisto por google [18], en donde se destacan 8 pautas:

- Hosteado en https
- Usa diseño adaptativo y se visualiza correctamente en mobile y tablets
- Todas las páginas deben funcionar cuando no se tiene conexión
- Debe tener metadatos para dar la opción "Agregar a la pantalla de Inicio" lo que permite "instalar" la aplicación.
- Debe tener un inicio rápido aún en redes lentas (<10 seg en redes 3g)
- Crossbrowsing (visualizarse correctamente independientemente del navegador).
- Los cambios de página deben ser rápidos
- Cada página debe tener su url y si es una app de una sola página con distintas vistas se debe poder reconstruir.

Los componentes principales de una PWA son:

1. Archivo de Manifiesto
2. Service Worker
3. Almacenamiento Local
4. Notificaciones

En este artículo el foco de interés está puesto en el almacenamiento local y como optimizar el uso de la cache para evitar descargar contenido innecesario de la red.

5. Esquemas de Cache Existentes

Las PWA permiten realizar una aplicación completamente desconectada, pero si los datos cambian dicho cache debe ser actualizado. Por lo que es necesario estudiar los distintos enfoques para el manejo de cache estableciendo la mejor estrategia para cada tipo de aplicación. Los enfoques posibles son:

- Solo cache: Utilizado para páginas estáticas que no cambian y siempre serán levantadas de la cache
- Solo red: Por el contrario del enfoque anterior siempre se irá a buscar el contenido a la red, muy útil para contenidos dinámicos que siempre cambian.
- Cache y si falla red: Esquema combinado en el que se prioriza la cache y en caso de no encontrar allí el contenido se recurre a la red. Al recuperarlo adicionalmente puede ser cargado en cache para futuros accesos.
- Red y si falla cache: Esquema combinado en el que se prioriza la red y si el usuario no tuviese conectividad el contenido a mostrar es el de cache.
- Carrera entre cache y red: Con carrera se refiere a que se consulta en cache y red en simultáneo y lo que primero se consiga es lo que se muestra. Por ejemplo, si se cuenta con equipos de acceso lento al almacenamiento, pero con una excelente conexión a la red para recursos pequeños puede ser más rápida la red que la cache.
- Cache y después red: Se muestra al usuario rápidamente lo que contiene la cache y mientras tanto se busca en la red si hay actualización del contenido.
- Contenido de reserva: Si no está disponible en cache y no hay acceso a la red, tener contenido de reserva para mostrarle al usuario (página de sin conexión o indicación por defecto).

Es el service worker el que se ocupará de trabajar con el manejo de esquemas de cache o del acceso a la red. A modo de ejemplo se presenta la figura 6 (tomada de [19]), puede observarse que se recurre al service worker el cual accede a cache y si el contenido no está disponible accede a la red para buscarlo y poder servirlo. Este esquema prioriza la cache, con la posibilidad de actualizar la cache en los casos que se requiera acceder a la red, lo cual está identificado con la línea punteada (que se le ha agregado a la figura del autor). Priorizar la cache permite no descargar datos que ya se encuentran disponibles en el dispositivo.

En la sección siguiente se profundiza sobre el funcionamiento de service worker para poder luego en la sección 7 explicar detalladamente una propuesta de manejo de cache.

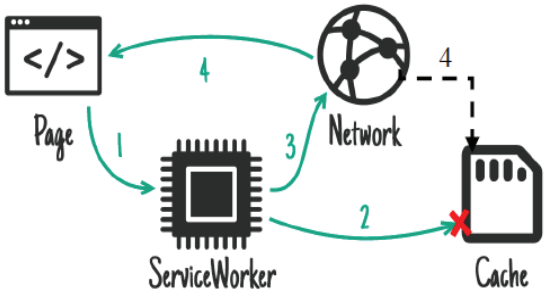

Figura 6. Esquema de acceso con prioridad a cache y luego red.

Un elemento central para el manejo de cache es la creación del serviceworker.

6. Funcionamiento de un Service Worker

Un service worker es un proceso que el navegador web ejecuta en segundo plano y está asociado a un sitio web particular. Este proceso se programa en javascript y permite capturar las peticiones que el sitio web hace a la red e interceptarlas actuando como un proxy local. El capturar esas peticiones permite que el service worker responda en lugar de la red, haciendo posible que el navegador no salga a la red sino que se le devuelvan los datos localmente.

El service worker tiene una serie de estados que permiten descargar datos de la red, atender clientes, mostrar notificaciones, etc. Cuando se accede a un sitio web construido con esta tecnología, un código javascript instala localmente el service worker en el navegador del cliente. El proceso de instalación permite hacer una descarga inicial de datos y almacenarlos localmente. Una vez instalado el service worker podrá comenzar a responder e interceptar las peticiones realizadas por el sitio web pudiendo retornar los datos almacenados localmente o ir a buscar a la red si es necesario y según sea el esquema preferido de recuperación de datos. El acceder a datos locales permite realizar aplicaciones que trabajen íntegramente con datos locales sin necesidad de acceso a la red. También reduce la necesidad de descargar páginas, imágenes y datos que no se actualizan frecuentemente haciendo que se descarguen una única vez y luego sean recuperadas de cache.

Ante un cambio del service worker se crea un nuevo proceso en el navegador conviviendo con el anterior pero aún no entrando en vigencia. La nueva versión entrará en vigencia recién cuando todos los clientes del service worker de la versión anterior se desconecten. Es importante aclarar que cada versión del service worker tiene acceso a datos locales propios por lo tanto es importante borrar la cache que ya no se utilizará al detectar que la nueva versión entra en vigencia. Este procedimiento se realiza mediante el evento activate del service worker.

7. Propuesta de Manejo de Cache

Se presentaron previamente los distintos esquemas de caché, estos esquemas pueden ser seleccionados dependiendo de las características de la aplicación. Por ejemplo, una aplicación que usa datos dinámicos los cuales siempre requieren ser actualizados obligatoriamente utilizará el esquema Solo Red, en contrapartida una aplicación que utiliza datos estáticos requerirá el esquema Sólo Cache. Luego en este último esquema planteado de sólo cache si las páginas no están disponibles se puede recurrir a la red trasladándonos al esquema “Cache y después Red”. Pero no hay estrategias mucho más complejas, se presenta la posibilidad de recurrir primero a la red o primero a cache o utilizar ambos viendo de cual se puede extraer primero los datos lo que supone un esquema de “carrera”. Es por ello que se plantea la necesidad de contemplar soluciones que tengan contenidos diversos y requieran de combinar los esquemas de cache, lo cual es explicado a continuación.

Al momento de desarrollar sitios web dinámicos que muestren, consulten y actualicen datos de un servidor existen dos esquemas posibles:

1. Que el servidor “arme” las páginas web y las devuelva al cliente en cada petición con los datos que se deben visualizar. Este esquema hace que todo el procesamiento se encuentre del lado del servidor haciendo que el cliente solo visualice los datos y pueda enviar actualizaciones mediante formularios.
2. El segundo esquema es que el cliente (navegador web) tome protagonismo y sea responsable de gran parte del procesamiento. Es decir que al recuperar una página web esta no es devuelta completa por parte del servidor, sino que los datos dinámicos son recuperados luego, mediante javascript y peticiones posteriores del cliente al servidor.

Nuestra propuesta se basa en la utilización de este segundo esquema ya que para poder utilizar una PWA indefectiblemente se necesita de un equipo con capacidad de procesamiento y no solo de visualización.

Es posible dividir la propuesta en 4 partes

- Estructura del sitio web
- Acceso a datos remotos
- Actualización de datos
- Manejo de versiones

La figura 7 muestra un resumen de cada una de las partes de la propuesta, las cuales serán explicadas a continuación.

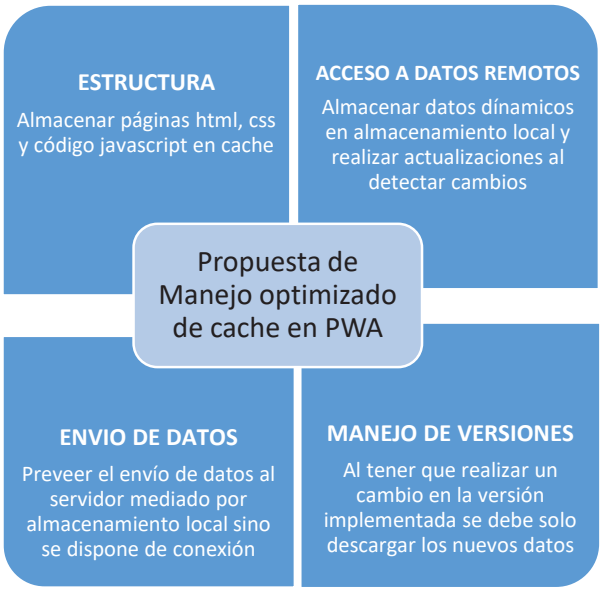

Figura 7. Resumen de la propuesta para la utilización de PWA con un uso optimizado de cache

7.1. Estructura del sitio web

Se debe desarrollar el sitio web de forma que los datos dinámicos sean recuperados exclusivamente mediante javascript.

Utilizar el poder de procesamiento del cliente para poder recuperar los datos trae la ventaja de que las páginas web recuperadas del servidor se transformen en página estáticas ya que solo contendrán elementos estructurales junto con el código JavaScript para la comunicación con el servidor. Dicho código permite recuperar datos dinámicos y procesarlos localmente para realizar el llenado de los controles en la página que será visualizada por el usuario. Es decir que el html descargado no cambia entre solicitud y solicitud. Los datos dinámicos son actualizados por el código incrustado en cada página.

Por lo tanto al instalar la PWA se debe guardar en cache:

- Páginas web
- Imágenes decorativas o de uso frecuente
- Hojas de estilo css
- Código Javascript

El utilizar una PWA permite sacar provecho de este esquema de trabajo de la siguiente manera:

1- Como las páginas html, imágenes y hojas de estilo no varían se pueden guardar en cache evitando tener que recuperarlas en cada request.
2- El código javascript tampoco cambia, solo se modificará ante un cambio en la versión del sistema.
3- El acceso dinámico mediante javascript a los datos del servidor puede ser interceptado por el service worker y recuperar datos previamente almacenados en cache

7.2. Acceso a datos remotos

Cuando sea necesario recurrir al servidor para recuperar datos si estos no cambiaron se recuperan del almacenamiento local.

Debido a que el service worker actúa como un proxy local, toda solicitud al servidor puede ser interceptada.

Desde el service worker es posible detectar cuando desde javascript se están solicitando datos al servidor y responder en lugar del servidor, con datos locales en lugar de acceder a la red. Estos datos locales pueden almacenarse utilizando algunos de los esquemas de almacenamiento local disponible. Existen diversas APIs provistas por el W3C (consorcio web internacional) por ejemplo: Web Storage [20], Indexed Database [21].

Una posible solución para minimizar el acceso a la red y reducir el consumo de datos es la siguiente:

1- Cuando se detecta una solicitud de datos que es interceptada por el service worker, se consulta a una base de datos local si están esos datos y una tabla separada se guarda la última fecha de actualización.
2- Se realiza una consulta al servidor para que nos devuelva si esos datos han cambiado. Para ello es necesario que el servidor guarde un registro de modificación de los datos que se recuperan para poder retornar la fecha de última actualización.
3- Si la fecha de actualización del servidor es más nueva que los datos locales, entonces se recuperan los datos y se almacenan localmente al mismo tiempo que son enviados a la página que los solicitó. En cambio, si los datos locales tienen la misma fecha que los del servidor se envían a la página los datos locales sin necesidad de recuperar nuevamente toda la información reduciendo considerablemente el tráfico de la red.

Este esquema si bien requiere una consulta adicional al servidor, en la mayoría de los casos serán muy pocos los datos que retorne (una fecha y hora) evitando recuperar nuevamente todos los datos cada vez que se accede a una página.

Por supuesto que en el primer ingreso los datos deben recuperarse, así como también se deben actualizar los datos locales cuando el cliente agregue nuevos datos guardándolos tanto en el servidor como en la base de datos local.

Esta metodología también permitirá el trabajo sin conexión mostrando datos al usuario disponibles en forma local, pudiendo detectar cuando esto ocurra e informar al usuario que los datos podrían no estar actualizados.

7.3. Envío de datos

Para que la aplicación sea funcional sin disponer de conexión también es posible agregar funcionalidad de

guardado local de nuevos datos para poder luego sincronizarlos con el servidor al volver a tener conexión.

Esta funcionalidad también se puede incorporar en el service worker al interceptar un envío de datos al servidor. En el service worker se puede chequear si está disponible el acceso a la red o no. Si está disponible se actualizan los datos en el servidor y en la cache local. Pero sino está disponible, los datos se almacenan solo localmente y se marcan como pendiente de sincronización. Luego en futuras solicitudes al detectar que existe conexión a internet se pueden enviar los datos para actualizar el servidor.

Este enfoque si bien es posible, debe tener mayores consideraciones para evitar persistir datos incorrectos, desactualizados o irrelevante al momento de volver a tener conexión. Siempre es importante informar al usuario lo que está pasando, avisarle cuando no hay conexión y cuando se intenta sincronizar los datos para que no dé por sentado que sus datos están guardados en el servidor cuando en realidad no lo están.

7.4. Manejo de versiones

Al trabajar de la forma planteada los elementos estructurales solo cambiarán si se quiere implementar una nueva versión de la aplicación. Como por ejemplo:

- Cambios en la forma de recuperación y/o procesamiento de datos (código javascript)
- Cambios de organización de componentes del sitio (páginas web)
- Cambios en la visualización, colores, tipografías, tamaños (hojas de estilo)
- Cambios gráficos en las imágenes utilizadas frecuentemente

Todos estos cambios pueden significar el modificar alguno/s de los recursos estructurales de la aplicación que se encuentran almacenados en cache y por lo tanto no se vuelven a descargar del servidor.

Para poder actualizar un recurso en la cache local se debe modificar el archivo del service worker en el servidor. Cuando el cliente detecta que este archivo fue modificado entiende que existe una nueva versión y procede a su instalación haciendo que la versión anterior ya no se utilice para nuevas instancias. Esta nueva versión entrará en vigencia recién cuando todos los clientes locales que están usando la versión anterior de service worker se desconecten, pero existen formas de detectar esa actualización y forzar a que se utilice la nueva versión.

Se utiliza el método “install” del service worker para recuperar las páginas del servidor y copiarlas a la cache local. El problema de este método es que lo que se detecta es un cambio en service worker y no en las páginas individuales entonces todas las páginas configuradas para ser recuperadas del servidor se van a sobrescribir sin importar si fueron modificadas o no. Cada service worker maneja su propio cache. Esto no debería ser un problema mayor ya que no es habitual realizar cambios muy seguido en el comportamiento de un sistema una vez que está probado e implementado.

Sin embargo, existe una forma de evitar descargar toda la información nuevamente ante un cambio de versión. En el service worker se definen dos colecciones de páginas, una colección con páginas que se actualizaron y deben descargarse y otra colección con las páginas que pueden copiarse de la cache anterior. La figura 8 muestra un ejemplo de un archivo de service worker donde la primera línea indica el número de versión y luego define las dos colecciones: paginasModificadas y recursosACopiar.

```
const cacheActual = 'miSitio-v2';

const paginasModificadas = [
  'grabar.html',
  'listado.html'
];

const recursosACopiar = [
  'css/estilos.css',
  'icons/home.svg',
  'home.html',
  'menu.html'
];
```

Figura 8. Ejemplo de archivo de service worker

Luego en el método install se recorren ambas colecciones copiando las páginas que no se modificaron y recuperando del servidor las modificadas. El código del evento install puede verse en la figura 9.

```
self.addEventListener("install",
function(event) {
  event.waitUntil(
caches.open(cacheActual).then(functio
n(cache) {
      var newImmutableRequests = [];
      return Promise.all(
recursosACopiar.map(function(url) {
          return
caches.match(url).then(function(respo
nse) {
            if (response) {
              return cache.put(url,
response);
            } else {
newImmutableRequests.push(url);
              return
Promise.resolve();
            }
          });
        })
      ).then(function() {
        return
cache.addAll(newImmutableRequests.con
cat(paginasModificadas));
      });
    })
  );
});
```

Figura 9. Metodo Install

Por último, debido a que cada service worker maneja su propia cache, cuando entra en vigencia la nueva versión, debe eliminarse la cache anterior para que en el cliente no queden datos obsoletos, para ello se agrega el código de la figura 10 en el método activate donde la última línea de código hace además que si todavía hay clientes vigentes se actualicen para utilizar esta nueva versión.

```
self.addEventListener("activate",
function(event) {
  event.waitUntil(

caches.keys().then(function(cacheNames)
{
      return Promise.all(

cacheNames.map(function(cacheVieja) {
     if (cacheVieja !== cacheActual)
     {
      return caches.delete(cacheVieja);
     }
    })
   );
  })
 );
  return self.clients.claim();
});
```

Figura 10. Manejo de diversos service workers

Para que esta metodología pueda ser empleada se debe actualizar el archivo del service worker con los cambios cada vez que se implemente una nueva versión. Esto reduce considerablemente el tiempo de instalación de una nueva versión y reduce considerablemente el consumo de datos de red ya que en general son pocas las páginas que se actualizan. Pero surge un nuevo problema a considerar, si en cada versión que se implementa solo se ponen como recursos nuevos las páginas modificadas entonces aquellos clientes que se salteen alguna actualización no van a descargar todas las páginas que tiene desactualizadas sino solo los de la última actualización.

Para evitar el problema de que los clientes queden parcialmente actualizados existen dos formas:

1- En cada actualización no quitar de la colección de páginas a modicar las páginas modificadas en versiones anteriores para que siempre descargue todo lo que difiera de la primer versión. Este esquema es funcional pero va a hacer que muchos usuarios que tengan al día sus versiones descarguen datos de más en actualizaciones posteriores.

2- Agregar lógica al método install, guardando un versionado del service worker en forma local y haciendo que, si se saltea una versión, por ejemplo si el service worker nuevo tiene la versión 3 pero localmente está instalada la 1 en ese caso como la versión 2 no fue descargada entonces se vuelva a descargar todo el sitio. Esto solo penaliza al usuario que no actualizó a tiempo su versión haciendo que el resto solo descarguen lo que sea necesario. Este versionado puede manejarse como una variable dentro del service worker (ver la primera línea de la figura 8) donde se debe ir incrementando en forma secuencial el número de versión ante cada actualización.

8. Conclusiones y Trabajos Futuros

La utilización de la metodología propuesta permite reducir la descarga innecesaria de información de la red, haciendo que solo la información nueva o que cambia sea descargada, tanto datos como páginas e imágenes. Se logra reducir el tráfico de la red y por consiguiente también el gasto que podría tener un usuario móvil al acceder con datos móviles a una aplicación web. El porcentaje de ahorro podrá variar entre los distintos tipos de aplicaciones según la cantidad de datos dinámicos y la frecuencia de actualización, pero al solo acceder al servidor para recuperar estos datos y ningún recurso adicional es una reducción más que considerable para cualquier aplicación web. También el esquema planteado reduce los tiempos de actualización de versiones ya que hace que no deba descargar nuevamente toda la aplicación, mejorando la experiencia del usuario, evitando demoras innecesarias al querer usar la aplicación y tener que esperar que se instale la nueva versión.

Algunas pautas simples de implementar, como el número de versionado ya permiten una importante mejora, si a su vez se puede trabajar con esquemas de cache más elaborados como el propuesto en el presente artículo la mejora será aún más significativa. Una de las características primordiales de las PWA es la optimización de cache y es un campo de desarrollo muy interesante en donde hay mucho por trabajar. Como fue posible visibilizar en este artículo las PWA tienen diversas características que permiten una única solución que sea multiplataforma y además aprovechando las ventajas que se podría tener con una aplicación nativa (ícono, poderse utilizar de forma desconectada, acceso al hardware), es por ello por lo que es un campo muy importante de trabajo.

Referencias

[1] We Are Social Inc. “Digital around the world in april 2020”. https://wearesocial.com/blog/2020/04/digital-around-the-world-in-april-2020

[2] Castro Rojas, S. "Ubicuidad y comunicación: los Smartphones." Chasqui: Revista Latinoamericana de Comunicación 118 (2012): 91-95.

[3] Fortunato, D., and Bernardino J. "Progressive web apps: An alternative to the native mobile Apps." 2018 13th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2018.

[4] Statcounter. “Mobile Operating System Market Share Worldwide”. 2020

[5] Fortunato, D., & Bernardino, J. (2018, June). Progressive web apps: An alternative to the native mobile Apps. In 2018 13th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). IEEE.

[6] Nurwanto, N. (2019). Penerapan Progressive Web Application (PWA) pada E-Commerce. Techno. Com, 18(3), 227-235.

[7] Aminudin, A., Basren, B., & Nuryasin, I. (2019). Perancangan Sistem Repositori Tugas Akhir Menggunakan Progressive Web App (PWA). Techno. Com, 18(2), 154-165.

[8] Nugroho, L. E., Pratama, A. G. H., Mustika, I. W., & Ferdiana, R. (2017, October). Development of monitoring system for smart farming using Progressive Web App. In 2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE) (pp. 1-5). IEEE.

[9] Kiswanto, N. P., Paturusi, S. D., & Tulenan, V. (2020). Aplikasi E-Log Book Penangkapan Ikan Menggunakan Progressive Web App. Jurnal Teknik Informatika, 15(2), 93-100.

[10] Rêgo, F., Portela, F., & Santos, M. F. (2019). Towards PWA in Healthcare. Procedia Computer Science, 160, 678-683.

[11] Gambhir, A., & Raj, G. (2018, June). Analysis of cache in service worker and performance scoring of progressive web application. In 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE) (pp. 294-299). IEEE.

[12] González-Bañales, D. L., & Monárrez Armendáriz, C. (2015). "Aplicación de principios de diseño adaptativo para el acceso a la plataforma Moodle en dispositivos móviles".

[13] Martínez, R., Rodríguez R., and Vera P. "Análisis del diseño adaptativo en sitios web gubernamentales." XXIV Congreso Argentino de Ciencias de la Computación (La Plata, 2018). 2018.

[14] Rodríguez Pérez, P. "Desarrollo de un cliente web mediante aplicaciones web progresivas". Universidad de Vigo. 2018 http://castor.det.uvigo.es:8080/xmlui/bitstream/handle/123456789/224/TFG%20Pablo%20Rodr%C3%ADguez%20P%C3%A9rez.pdf?sequence=1

[15] Gallagher N., "How We Built Twitter Lite," Apr. 2017.

[16] Adetunji, O., Ajaegbu, C., Otuneme, N., & Omotosho, O. J. (2020). "Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development". American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), 68(1), 85-99.

[17] Rodríguez, R. A., Vera, P. M., Martínez, R., Parra Beltrán, F., Trigueros, A., & Dogliotti, M. "Aplicaciones web progresivas impulsadas por el avance de los estándares web. In XXI Workshop de Investigadores en Ciencias de la Computación (WICC 2019, Universidad Nacional de San Juan).

[18] Google Developers. "Progressive Web Apps" https:/developers.google.com/web/progressive-web-apps/

[19] Archibald J. "La guía de soluciones sin conexión". Google Developers https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/?hl=es

[20] W3C. "Web Storage (Second Edition)". 2016 https://www.w3.org/TR/webstorage/

[21] W3C. "Indexed Database API 2.0". 2018 https://www.w3.org/TR/IndexedDB-2/