TecnoLógicas

ISSN-p: 0123-7799
ISSN-e: 2256-5337

Vol. 24, nro. 51, e1892, 2021

Recibido: 28 febrero 2021
Aceptado: 21 junio 2021
Disponible: 12 julio 2021

Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva

Comparative Analysis of the Performance Efficiency of a Traditional vs a Progressive Web Application

Jhonatan Llamuca-Quinaloa[1];
Yasmani Vera-Vincent[2];
Verónica Tapia-Cerda[3]

[1] Universidad Técnica de Cotopaxi, Latacunga-Ecuador, jhonatan.llamuca5258@utc.edu.ec
[2] Universidad Técnica de Cotopaxi, Latacunga-Ecuador, yasmani.vera1583@utc.edu.ec
[3] Universidad Técnica de Cotopaxi, Latacunga-Ecuador, veronica.tapia@utc.edu.ec

Cómo citar / How to cite

J. Llamuca-Quinaloa; Y. Vera-Vincent; V. Tapia-Cerda, "Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva", *TecnoLógicas*, vol. 24, nro. 51, e1892, 2021. https://doi.org/10.22430/22565337.1892

Resumen

El desarrollo de esta investigación tuvo por objetivo comparar dos tipos de tecnología de aplicaciones web: aplicación web tradicional y aplicación web progresiva, para determinar el tipo de tecnología que tiene un mejor desempeño con respecto al atributo de calidad de rendimiento. Para este fin, primero se desarrolló una aplicación web tradicional, para luego configurarla y convertirla en una aplicación web progresiva; de esta manera se obtuvieron dos tipos de aplicaciones que permitieron medir su cualidad de rendimiento. El análisis comparativo utiliza el estándar ISO/IEC 25010 para la elección de la característica de eficiencia de desempeño (rendimiento), el estándar ISO/IEC 25040, establece un proceso de cinco actividades para evaluar cada aplicación, y el estándar ISO/IEC 25023, define las características de rendimiento con sus respectivas métricas y fórmulas. El estudio también mostró el comportamiento que tienen las aplicaciones en navegadores compatibles con trabajadores de servicio: Chrome, Firefox, Opera y Edge. Las herramientas utilizadas para medir las métricas de rendimiento fueron: page load time, load time, el administrador de tareas y las herramientas de desarrollo, todas estas integradas en los navegadores. Los resultados mostraron discrepancias en el comportamiento de las aplicaciones, pero fueron similares en métricas como: comportamiento en el tiempo y la capacidad del software. Con respecto a las métricas de utilización de recursos, la aplicación web progresiva presentó un elevado consumo. Sin embargo, esta aplicación brinda una mejor experiencia al usuario, pero el almacenamiento en caché y el soporte de los navegadores puede ser una limitante para su correcto funcionamiento.

Palabras clave

Aplicación web progresiva, aplicación web tradicional, análisis comparativo, trabajador de servicio, almacenamiento en caché.

Abstract

This study compares two types of web application (traditional and progressive) to determine the kind of technology that achieves a better performance quality. For that purpose, a traditional web application was developed, then configured and converted into a progressive web application. As a result, two types of application were obtained in order to measure their performance quality. This comparative analysis used the ISO/IEC 25010 standard to select the performance efficiency characteristic. In addition, it implemented the ISO/IEC 25040 standard, which establishes a process of five activities to evaluate each application, and the ISO/IEC 25023 standard, which defines performance characteristics with their respective metrics and formulas. This paper also examines the behavior of the applications in browsers compatible with service workers: Chrome, Firefox, Opera, and Edge. Several tools were used to measure the performance metrics: page load time, load time, the task manager, and development tools (all of them integrated into the browsers). The results show discrepancies in the behavior of the applications, although they presented similar metrics in terms of behavior over time and software capacity. Regarding resource utilization metrics, the progressive web application presented a higher consumption. This type of application provides a better user experience but caching and browser support can be limitations for its correct operation.

Keywords

Progressive web application, traditional web application, comparative analysis, service worker, cache storage.

1. INTRODUCCIÓN

Entre 1990 y 1995, los sitios web eran un conjunto de archivos de hipertexto vinculados con contenido de texto y gráfico limitado [1]. Internet es una de las tecnologías recientes que más rápido se está desarrollando [2]. Con su crecimiento, los sitios web se han expandido en diferentes áreas, generando soluciones efectivas ante diferentes problemáticas.

Las páginas web, a lo largo del tiempo han evolucionado, facilitando a las personas realizar diversas tareas cruciales como vender y comprar bienes, realizar tareas breves, controlar casas inteligentes y administrar cuentas bancarias [3]. Para 2020 había más de 1800 millones de sitios web en internet [4]. Esta evolución también se ha asociado a la aparición de nuevos lenguajes de programación, herramientas y metodologías para el desarrollo de aplicaciones web [2]. La evolución de la web es un claro ejemplo de cómo la sociedad ha progresado con el uso de la tecnología, pasando de visualizar información estática a interactuar con sitios web dinámicos.

Una aplicación web (web-based application) es un tipo especial de aplicación cliente/servidor [5]. Funcionan bajo un navegador con conexión a Internet, ya que deben comunicarse con un servidor para cumplir con sus funcionalidades. Las aplicaciones web utilizan tres tecnologías básicas para su desarrollo: HTML, CSS y JavaScript.

La evolución constante de nuevas tecnologías en las páginas web ha ido creando diversas formas más sencillas, eficientes y seguras de navegar, brindando una mejor experiencia para el usuario. Recientemente ha surgido una nueva alternativa para el desarrollo de aplicaciones web móviles denominada aplicaciones web progresivas (PWA, por sus siglas en inglés) [6].

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 [7]. Esta tecnología funciona del lado del cliente y utiliza la interfaz de programación de aplicaciones (API, por sus siglas en inglés) para conectarse a diferentes recursos de los dispositivos. Utilizan el protocolo HTTPS y no dependen de la conectividad a Internet, ya que el navegador ejecuta los *service workers* en segundo plano, permitiendo que las aplicaciones funcionen en redes de baja calidad o incluso sin conectividad [8]. Es necesario tener una conexión a Internet la primera vez que se instala la aplicación y haber navegado previamente en la misma para que se puedan registrar los *service workers*.

Para evaluar un producto *software* es necesario seguir un modelo de evaluación que permita medir su nivel de calidad. La ISO/IEC 25040 es un modelo de referencia para la evaluación [9] que permitió evaluar las dos tecnologías mediante un proceso de cinco actividades. Por su parte, la ISO/IEC 25023 permite medir cuantitativamente la calidad de un *software* [10] a través de un conjunto de métricas que se basan en la ISO/IEC 25010 [11]. Las métricas utilizadas forman parte de la característica de calidad del rendimiento, las cuales son: comportamiento en el tiempo, utilización de recursos y capacidad [12].

La evaluación también se ejecuta en navegadores compatibles con trabajadores de servicio: Chrome, Firefox, Opera y Edge. Es importante tener en cuenta diferentes escenarios para realizar pruebas de rendimiento, ya que es probable que los usuarios interactúen de forma diferente con sitios web y aplicaciones en su vida cotidiana [13]. Este tipo de estudio comparativo permite obtener resultados que muestran las diferencias en cómo un usuario utiliza una aplicación y cómo esta le responde [13]; además, estos resultados muestran la importancia que tiene el uso adecuado del almacenamiento en caché y la correcta configuración de los trabajadores de servicio para que la PWA pueda tener un rendimiento eficiente.

Para la evaluación de las métricas de rendimiento se utilizaron herramientas integradas en cada navegador, estas herramientas son: page load time en Chrome, Opera y Edge, y load

time en Firefox utilizados para medir las métricas de comportamiento en el tiempo; el administrador de tareas para verificar la utilización de recursos de cada aplicación; y las herramientas de desarrollo para verificar el almacenamiento en caché de la PWA y el comportamiento que tienen las aplicaciones a través de la red.

El presente estudio evalúa dos versiones de una aplicación web para *delivery*: una aplicación web tradicional y una aplicación web progresiva. Se utilizó el modelo de desarrollo iterativo-incremental para agilizar los procesos de desarrollo y obtener las dos versiones propuestas. Además, se muestran las arquitecturas de estas aplicaciones, la configuración de la PWA, el proceso utilizado por los trabajadores de servicio para almacenar y responder con información almacenada en la caché y los navegadores, y sistemas operativos compatibles con esta tecnología.

2. METODOLOGÍA

El proceso metodológico se inicia con la consulta de fuentes bibliográficas que ayudaron a definir un modelo de evaluación para productos de *software* y establecer el rendimiento como único atributo de calidad a evaluar. Además, ayudó a definir las métricas de rendimiento con sus respectivas fórmulas y las herramientas informáticas de evaluación que permitieron generar los resultados comparativos.

Este estudio permite verificar el desempeño que tienen las aplicaciones con base en pruebas que monitorean su funcionamiento y dan a conocer las similitudes, diferencias o discrepancias que existen al compararlas. Una de las importancias que tiene el estudio es que obtiene resultados relevantes que permiten a otros investigadores usarlos para sus respectivas investigaciones.

En los estudios, [14]-[16] realizaron un análisis de las características de las PWA para unificar el desarrollo de las aplicaciones nativas, híbridas, móviles y web, sus análisis se enfocaron en la portabilidad, usabilidad y accesibilidad, sin embargo, no tomaron muy en cuenta el rendimiento; por tal motivo, este estudio comparativo se centra únicamente en el rendimiento de aplicaciones web para demostrar el desempeño que tienen las aplicaciones bajo estas métricas.

Este análisis comparativo utiliza la familia de normas ISO/IEC 25000 para establecer la característica de calidad a evaluar en cada aplicación y el modelo de referencia para la evaluación; su finalidad es garantizar la calidad de un producto *software* con base en un marco de trabajo común [17].

El estándar ISO/IEC 25040 establece un proceso de cinco actividades [9] que fueron tomadas en cuenta para establecer las métricas de evaluación, describir el proceso de configuración de la PWA, identificar el comportamiento de la aplicación con el uso de herramientas de *software* y obtener resultados que contribuyan a la toma de decisiones de desarrolladores web.

2.1 Métricas para evaluar el rendimiento de aplicaciones

El análisis comparativo utiliza el estándar ISO/IEC 25010 que describe ocho características para evaluar el *software* [11], para lo cual se utilizó solo una característica de calidad: eficiencia de desempeño (rendimiento). Los motivos se presentan a continuación:

Una aplicación web tiene diferentes atributos y métricas que debe seguir para ser considerado un producto de calidad, por lo que no es posible abarcar todos los atributos para un sistema [18] en una sola evaluación.

En [18] se menciona que, según una encuesta elaborada por Capgemini, Sogeti y Hewlett Packard (HP), el rendimiento es uno de los atributos de calidad más importantes, y este debe ser considerado en una evaluación de *software*, ya que permite medir la eficiencia que tiene al interactuar con los usuarios. En la Tabla 1 se observan las subcaracterísticas de rendimiento y sus respectivas métricas.

Tabla 1. Subcaracterísticas de rendimiento con sus respectivas métricas. Fuente: [11],[19].

Subcaracterística de rendimiento	Métricas
Comportamiento en el tiempo	Tiempo de respuesta
	Tiempo de espera
	Rendimiento
Utilización de recursos	Utilización de CPU
	Utilización de memoria
	Utilización de dispositivos de Entrada y Salida (E/S)
Capacidad	Número de peticiones en línea
	Número de accesos simultáneos
	Sistemas de transmisión de ancho de banda

2.2 Descripción del producto *software*

La aplicación web que hace posible el presente estudio comparativo, consiste en una plataforma web para *delivery* llamada VeciEntrega. Las tecnologías utilizadas para su desarrollo fueron: el *framework* de desarrollo web Django, el sistema gestor de base de datos PostgreSQL, HTML, CSS, JavaScript y tecnología PWA para una nueva versión de la aplicación. En la Figura 1 se observa la versión de la aplicación web tradicional y en la Figura 2, la versión de la PWA.

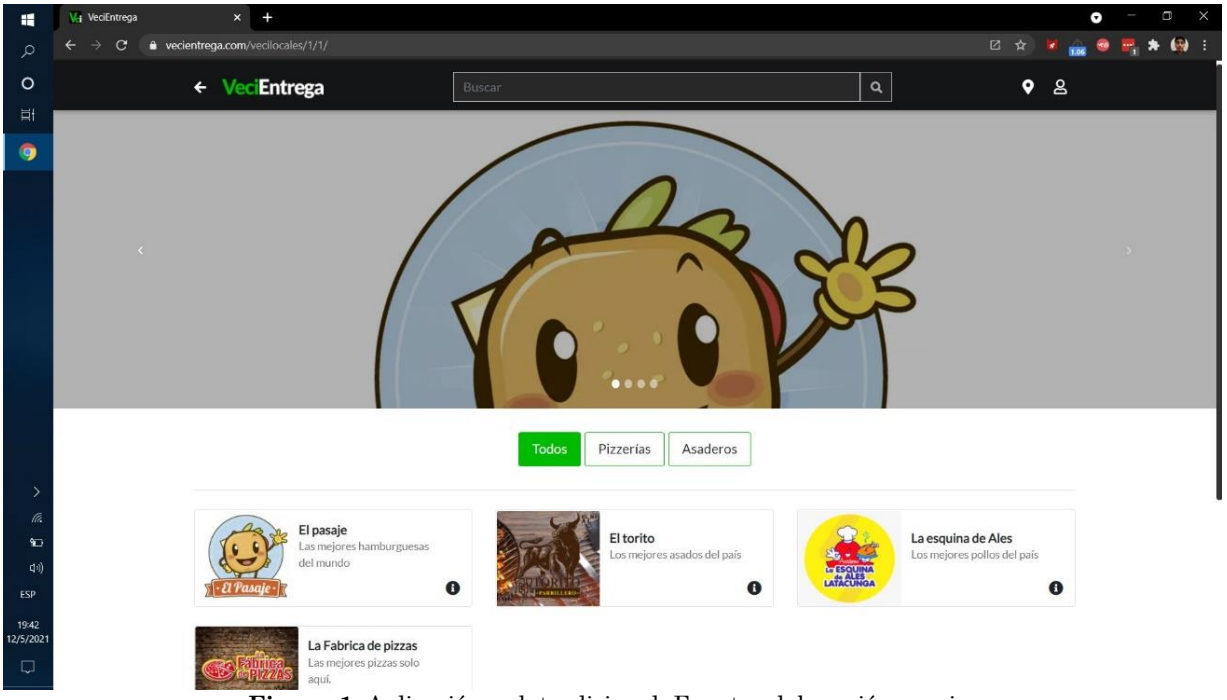

Figura 1. Aplicación web tradicional. Fuente: elaboración propia.

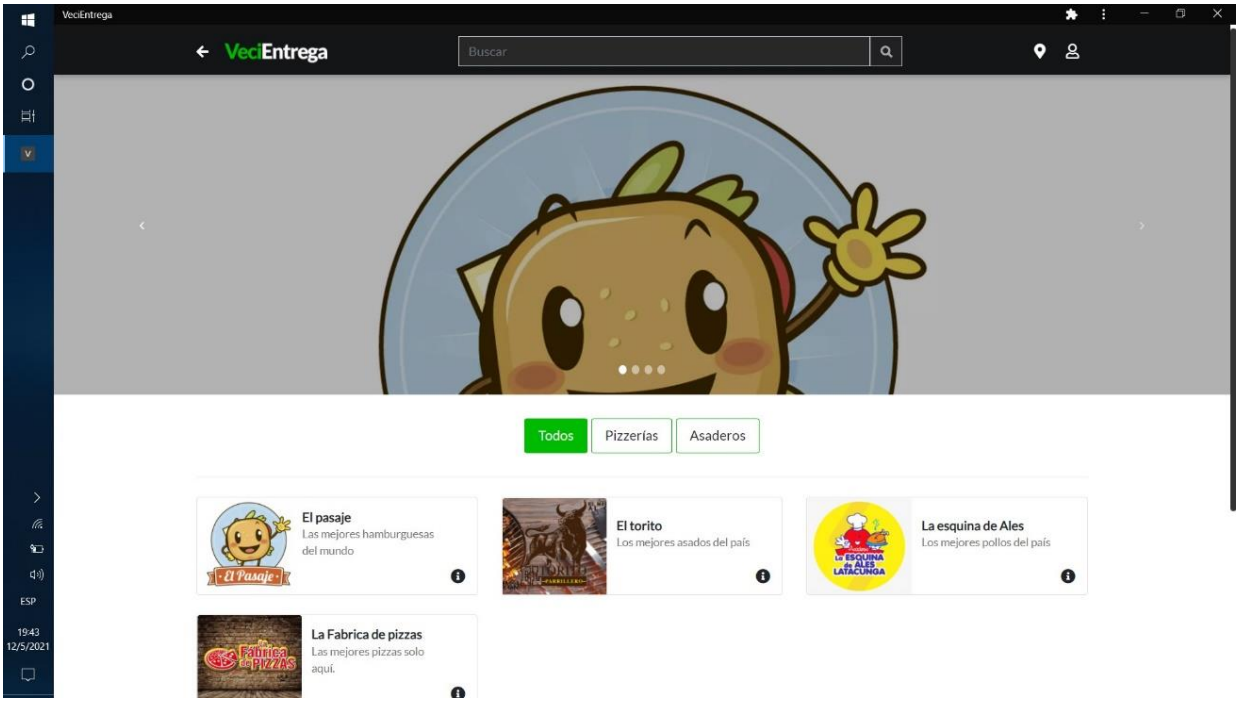

Figura 2. Aplicación web progresiva. Fuente: elaboración propia.

Se optó por desarrollar este tipo de aplicación debido a que durante la COVID-19 las personas empezaron a utilizar con mayor frecuencia canales electrónicos por motivos como: temor de contagiarse y restricciones de movilidad. En [20] se presenta un estudio realizado en Ecuador vinculado al comportamiento de los canales de compra, donde las aplicaciones de adquisición y *delivery* ocupan el segundo lugar con un 44 %

Al ser un tipo de aplicación de mayor auge, este debe presentar un alto rendimiento a la hora de atender las peticiones del usuario [21], al cumplir con esta característica el resultado será una aplicación más fluida y con tiempos de acceso menores.

2.3 Arquitectura de una aplicación web tradicional y una PWA

Una aplicación web tradicional debe tener una conexión a Internet para realizar peticiones y obtener respuestas por parte del servidor web; el tiempo de respuesta dependerá de la velocidad de internet. En la Figura 3 se observa la arquitectura de este tipo de aplicaciones.

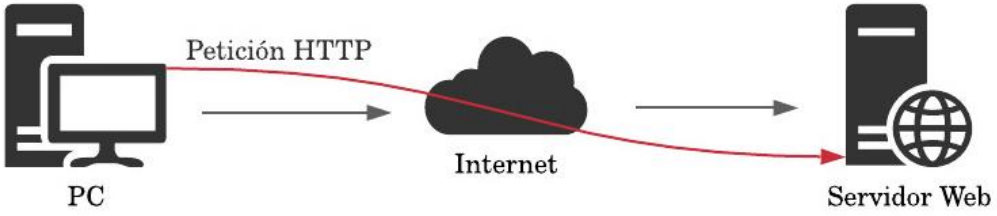

Figura 3. Arquitectura de una aplicación web tradicional. Fuente: elaboración propia.

Una PWA es una tecnología basada en la web tradicional que aprovecha recursos del navegador y de las API modernas para mejorar la experiencia del usuario. Entre sus características están: funcionamiento sin internet, actualizada, acceso a recursos del dispositivo e instalable, entre otros. La fórmula obtenida en [22] define la arquitectura de una PWA: HTTPS + trabajador de servicio + manifiesto de aplicación web = aplicación web progresiva.

El trabajador de servicio es un archivo de JavaScript de instrucciones que se ejecutan en segundo plano en el navegador, el cual permite almacenar datos en caché para seguir utilizando la aplicación sin internet. El manifiesto de aplicación web consiste en un conjunto de propiedades como: el nombre de la aplicación, el tema y el icono, entre otros, que indican a los navegadores web y a los dispositivos móviles cómo mostrar la aplicación. En la Figura 4 se observa la arquitectura de una PWA.

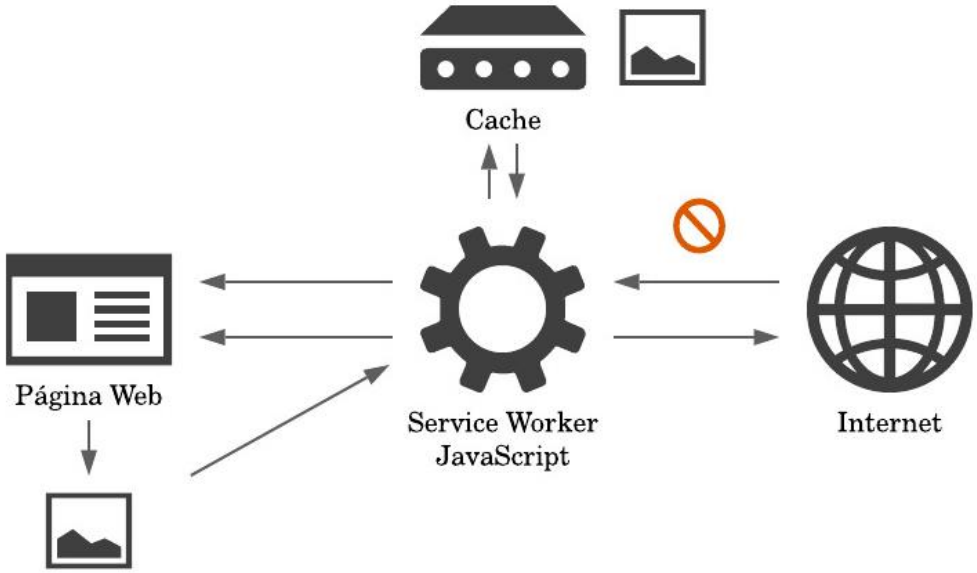

Figura 4. Arquitectura de una PWA. Fuente: elaboración propia.

2.4 Implementación de la PWA

El paquete utilizado para la implementación de una PWA en Django es: django-progressive-web-app==0.1.1. El archivo que intercepta las peticiones hechas por el cliente es el trabajador de servicio, y contiene código JavaScript que ejecuta instrucciones en segundo plano, este permitirá almacenar en caché: imágenes, código HTML, CSS y JavaScript. En la Figura 5 se muestra el fragmento de código que define el nombre y las URL que se guardan en caché.

Cuando el usuario accede por primera vez a la aplicación, los archivos que se guardan en caché son los presentes en la variable: urls_cache. El código de la Figura 6 contiene los eventos "install", que se encarga de instalar el trabajador de servicio, y "fetch", el cual se activa cuando se intenta recuperar archivos del caché, por lo que se encarga de interceptar, almacenar y responder.

serviceworker.js > self.addEventListener('install') callback

```
1  //Nombre de la cache
2  var nombre_cache = 'veciapp_cache_v1';
3
4  ///Url's que se almacenan en cache en la 1ra carga
5  var urls_cache = [
6      //Index
7      '/',
8      //Imagenes
9      '/static/styleCliente/img/index/banner001.jpg',
10     '/static/styleCliente/img/index/banner002.jpg',
11     '/static/styleCliente/img/index/banner003.jpg',
12     '/static/styleCliente/img/index/phone.png',
13     //Archivo JS
14     '/static/services/ciudades.js',
15     //Archivo Css
16     '/static/styleCliente/css/index.css',
17     '/static/styleCliente/css/generals.css',
18 ];
```

Figura 5. Archivos para almacenar en caché. Fuente: elaboración propia.

serviceworker.js > ...

```
20 //Instalación del service worker
21 self.addEventListener('install', function(event) {
22     event.waitUntil(
23         caches.open(nombre_cache)
24         .then(function(cache) {
25             console.log('Instalado');
26             return cache.addAll(urls_cache);
27         })
28     );
29 });
30
31 //Respuestas a las peticiones del usuario
32 self.addEventListener('fetch', function(event) {
33     event.respondWith(
34         fetch(event.request)
35         .then(function(result) {
36             return caches.open(nombre_cache)
37                 .then(function(c) {
38                     c.put(event.request.url, result.clone());
39                     return result;
40                 })
41         })
42         .catch(function(e) {
43             return caches.match(event.request)
44         })
45     );
46 });
```

Figura 6. Evento "install" y "fetch". Fuente: elaboración propia.

Ahora bien, existen archivos, como las imágenes, que son dinámicos y para mejorar el mantenimiento de la aplicación se agregó en el evento "fetch" el código que permite instalar trabajadores de servicio por cada página accedida y almacenar todos estos archivos en caché.

Finalmente, en la Figura 7 se muestra el archivo settings.py con la ruta que permite reconocer el trabajador de servicio y el código que representa al manifiesto de la aplicación.

```
veciapp > settings.py > ...
281  ##Ruta del trabajador de servicio
282  PWA_SERVICE_WORKER_PATH = os.path.join(BASE_DIR, 'serviceworker.js')
283
284  #Manifiesto de la aplicación
285  PWA_APP_NAME = 'VeciEntrega'
286  PWA_APP_SHORT_NAME = 'Pídelo Facilito, lo entregamos rapidito!'
287  PWA_APP_DESCRIPTION = "Aplicación Delivery"
288  PWA_APP_THEME_COLOR = 'rgb(10,10,10)'
289  PWA_APP_BACKGROUND_COLOR = 'rgb(0,0,0)'
290  PWA_APP_DISPLAY = 'standalone'
291  PWA_APP_SCOPE = '/'
292  PWA_APP_ORIENTATION = 'landscape'
293  PWA_APP_START_URL = '/'
294  PWA_APP_STATUS_BAR_COLOR = 'default'
295  PWA_APP_ICONS = [{ 'src': '../static/styleCliente/img/logos/logo.png', 'sizes': '250x250' }]
296  PWA_APP_ICONS_APPLE = [{ 'src': '../static/styleCliente/img/logos/logo.png', 'sizes': '250x250' }]
297  PWA_APP_SPLASH_SCREEN = [{ 'src': '../static/styleCliente/img/logos/logo.png', 'media': '(device-wi
298  PWA_APP_DIR = 'ltr'
299  PWA_APP_LANG = 'en-US'
```

Figura 7. Ruta del trabajador de servicio y manifiesto de la aplicación. Fuente: elaboración propia.

2.5 Navegadores compatibles con el trabajador de servicio

El trabajador de servicio es un archivo fundamental para integrar funcionalidades avanzadas a una PWA, sin embargo, no todos los navegadores soportan la instalación de estos archivos. En la Tabla 2 se observan los navegadores y los sistemas operativos compatibles.

Tabla 2. Navegadores y sistemas operativos compatibles con los trabajadores de servicio. Fuente: [23].

Navegador/ Sistema Operativo	Chrome	Safari	Edge	Firefox	Opera
Android	Sí	---	Sí	Sí	Sí
Windows	Sí	No	Pronto	Sí	Sí
iOS	No	Pronto	Pronto	No	---
OS X	Sí	No	---	Sí	Sí
Linux	Sí	---	---	Sí	Sí

Además, los navegadores integran un conjunto de herramientas para desarrolladores, estas son de gran utilidad para inspeccionar una aplicación web y verificar su comportamiento, almacenamiento, depuración y conexión, entre otros.

Para la evaluación se tomaron en cuenta cuatro navegadores compatibles con esta tecnología en Windows: Chrome, Firefox, Opera y Edge. Al incluir estos navegadores a la evaluación, se podrá identificar el comportamiento de la aplicación web tradicional y de la PWA bajo estos escenarios y obtener resultados que reflejen igualdad o discrepancias entre estos.

2.6 Almacenamiento en caché

Para ejecutar las pruebas de rendimiento, previamente se analizaron los archivos generados por la versión PWA. En la Figura 8 se muestra el panel donde se instalan y ejecutan los trabajadores de servicio.

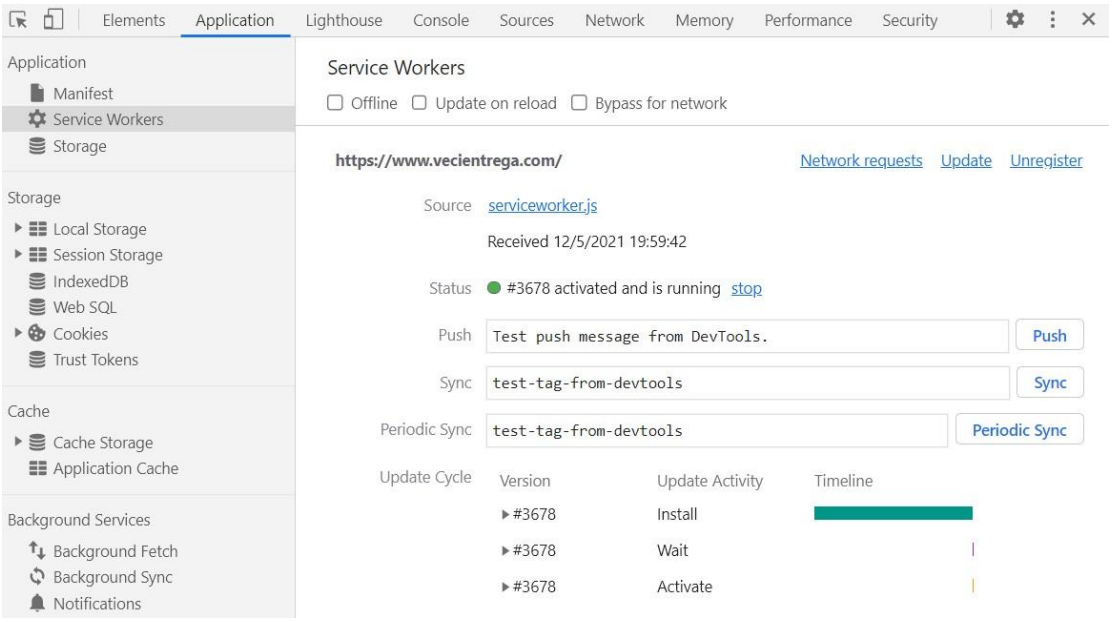

Figura 8. Panel de supervisión de trabajadores de servicio. Fuente: elaboración propia.

Este trabajador de servicio intercepta y guarda en caché todos los archivos provenientes del servidor, la configuración se la puede observar en la Figura 5. A medida que el usuario navega, el estado del trabajador de servicio pasará de instalado ha activado, ocasionando de esa manera que el usuario obtenga una respuesta del caché y utilice el servidor únicamente para realizar consultas a la base de datos.

En la Figura 9 se muestra el almacenamiento en caché, es aquí donde se guardan todos los archivos interceptados por el trabajador de servicio.

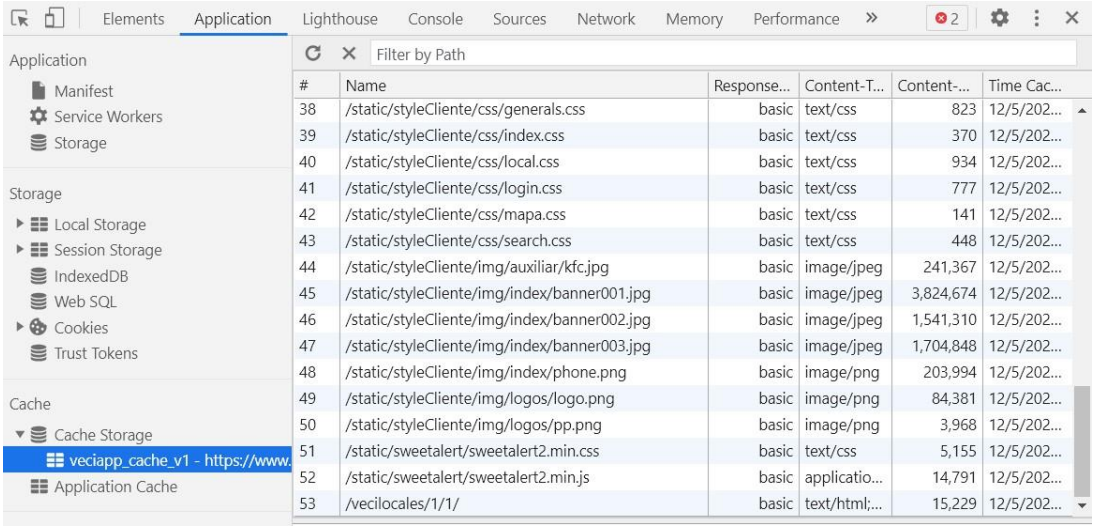

Figura 9. Almacenamiento en caché. Fuente: elaboración propia.

Cuando una aplicación almacena en caché es importante tener en cuenta los conceptos y las políticas del almacenamiento en caché web [24]. En la Figura 10 se muestra la sección de almacenamiento, que permite conocer la cuota de almacenamiento y el consumo actual de la aplicación.

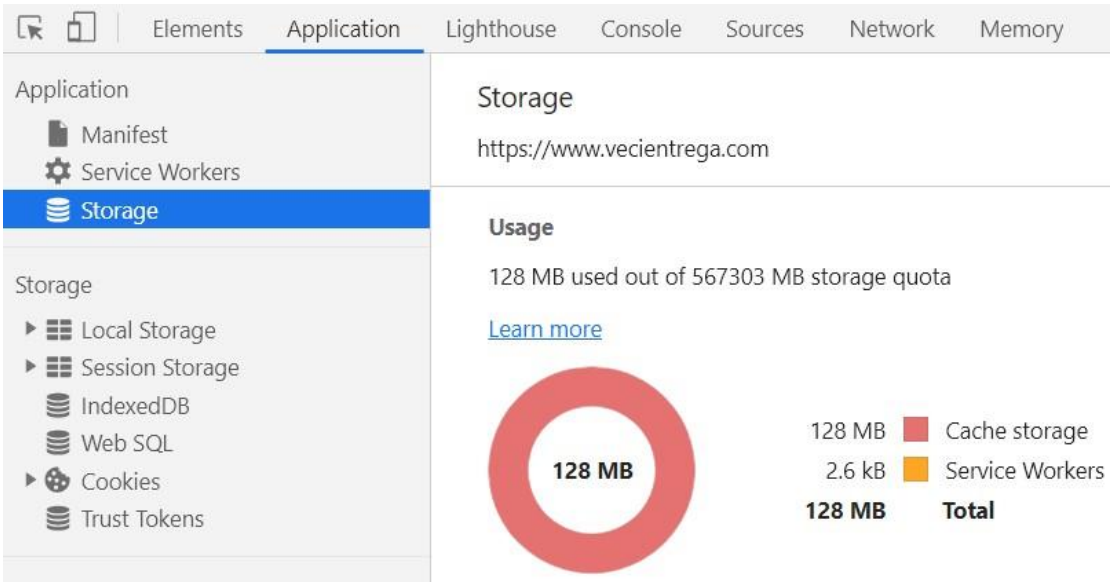

Figura 10. Capacidad y uso de almacenamiento. Fuente: elaboración propia.

El trabajador de servicio configurado para esta aplicación almacena en cachés archivos dinámicamente, por lo que el uso del almacenamiento en el navegador irá incrementando a medida que el usuario visite las páginas de la aplicación.

Los navegadores permiten diferentes cantidades de almacenamiento. En la Figura 11 se muestra un fragmento de código JavaScript para conocer estos datos.

```
//Ver disponibilidad de cache
navigator.storage.estimate().then(function(estimate) {
    var almac_total = (estimate.quota.toFixed(2)) / 1000000
    var almac_usado = (estimate.usage.toFixed(2)) / 1000000
    var almac_disponible = almac_total - almac_usado

    console.log("ALMACENAMIENTO TOTAL: " + almac_total + " MB.")
    console.log("ALMACENAMIENTO USADO: " + almac_usado + " MB.")
    console.log("ALMACENAMIENTO DISPONIBLE: " + almac_disponible + " MB.")
});
```

Figura 11. Valores del almacenamiento en caché. Fuente: elaboración propia.

2.7 Métricas y fórmulas para medir el rendimiento

Este tipo de evaluación se la puede realizar durante la etapa de pruebas de desarrollo de *software* [19]; pero, para tener un escenario y resultados confiables, se puso en producción a las dos versiones en Google App Engine. A continuación, se observan las métricas de rendimiento y las fórmulas que serán usadas para la evaluación.

2.7.1 Comportamiento en el tiempo

Estas métricas permiten medir los tiempos en que un *software* tarda en realizar una determinada actividad, desde que se inicia hasta que termina, incluyendo tiempos de prueba y operaciones [19]. En la Tabla 3 se observan estas métricas con sus respectivas fórmulas.

Tabla 3. Métricas y fórmulas de comportamiento en el tiempo. Fuente: [11][19].

Métrica	Fórmula	Descripción
Tiempo de respuesta	X = B-A	A = Tiempo de envío de petición. B = Tiempo en recibir la primera respuesta.
Tiempo de espera	X = B-A	A = Tiempo cuando se inicia un trabajo. B = Tiempo en completar el trabajo.
Rendimiento	X = A/T	A = Número de tareas completadas. T = Intervalo de tiempo.

Tiempo de respuesta y tiempo de espera: estos aspectos se consideraron ya que permiten conocer el comportamiento en el tiempo de la aplicación web para realizar una actividad. El tiempo de respuesta es la duración entre iniciar una tarea y recibir la primera respuesta, y el tiempo de espera es la duración entre iniciar y completar un trabajo [19].

Rendimiento: es el tiempo que le toma al *software* para completar una o varias tareas [19]. Esta métrica permite medir la eficiencia que tiene la aplicación para cumplir con sus actividades propuestas.

2.7.2 Utilización de recursos

Estas métricas permiten medir el comportamiento que tienen los recursos utilizados por el *software* [19]. En la Tabla 4 se observan estas métricas con sus respectivas fórmulas.

Tabla 4. Métricas y fórmulas de utilización de recursos. Fuente: [11][19].

Métrica	Fórmula	Descripción
Utilización de CPU	X = A	A = Cantidad de CPU que es usada para realizar una tarea.
Utilización de memoria	X = A	A = Cantidad de memoria que es usada para realizar una tarea.
Utilización de dispositivos de E/S	X = A/B	A = Tiempo de operación. B = La cantidad de dispositivo(s) de E/S ocupada para realizar una tarea.

Utilización de la CPU: este aspecto es muy importante debido a que la CPU se encarga de que las tareas se realicen a una mayor velocidad, por tal motivo su consumo en las páginas web debe ser mínimo para que las personas tengan una mejor experiencia de navegación.

Utilización de memoria: esta métrica permite asignar un espacio de memoria a un proceso. A medida que estos procesos van aumentando, el consumo de memoria de acceso aleatorio (RAM, por sus siglas en inglés) también aumenta.

Utilización de dispositivos de E/S: esta métrica permite medir el tiempo que consumen los dispositivos de E/S para realizar una determinada actividad [19].

2.7.3 Capacidad

Estas métricas permiten medir la capacidad de respuesta que tiene un *software* cuando es sometido a límites máximos de funcionamiento [19]. En la Tabla 5 se observan estas métricas con sus respectivas fórmulas.

Tabla 5. Métricas y fórmulas de capacidad. Fuente: [11][19].

Métrica	Fórmula	Descripción
Número de peticiones en línea	X = A/T	A = Número máximo de peticiones en línea procesadas. T = Tiempo de operación, T > 0
Número de accesos simultáneos	X = A/T	A = Número máximo de accesos simultáneos. T = Tiempo de operación, T > 0
Sistemas de transmisión de ancho de banda	X = A/T	A = Cantidad máxima de transmisión de datos. T = Tiempo de operación, T > 0

Número de peticiones en línea: permite medir las peticiones que son procesadas bajo un cierto intervalo de tiempo [19].

Número de accesos simultáneos: determina el número máximo de usuarios que acceden al sistema simultáneamente y el tiempo en que demora esta operación [19].

Sistema de transmisión de ancho de banda: mide el valor de transmisión máximo requerido para realizar las actividades en el sistema informático [19].

2.8 Herramientas informáticas para medir el rendimiento

Las herramientas informáticas utilizadas para la evaluación de las aplicaciones son: la página web MLAB, Novabench, la extensión page load time para Chrome, Opera y Edge, la extensión load time para Firefox, el administrador de tareas y las herramientas de desarrollo.

MLAB: este sitio web permite realizar pruebas de velocidad que proporciona diagnósticos avanzados del rendimiento de la conexión de banda ancha a través de mediciones rápidas. Se dedica a suministrar un ecosistema para la medición abierta y verificable del rendimiento de la red global [25].

Novabench: este *software* es un punto de referencia gratuito que permite realizar pruebas rápidas de rendimiento de componentes informáticos, como la unidad de procesamiento central (CPU), la unidad de procesamiento de gráficos (GPU) , transferencia de RAM y Speed (la velocidad de lectura y escritura del disco duro de la computadora) [26].

Extensión page load time y load time: estas extensiones miden el tiempo de carga de la página y lo muestran en la barra de herramientas [27]. Consiste en la cantidad de tiempo promedio que tarda una página en aparecer en la pantalla. Se calcula desde el inicio (cuando se hace clic en el enlace de una página o se escribe una dirección web) hasta su finalización (cuando la página está completamente cargada en el navegador). Normalmente, el tiempo de carga de la página es medido en segundos.

Administrador de tareas: esta herramienta permite conocer el consumo de recursos como: CPU, memoria y red de los sitios web que están siendo utilizados por el navegador.

Herramientas de desarrollo: este conjunto de herramientas se integra en los navegadores y permiten crear, depurar, evaluar y verificar el comportamiento de aplicaciones web en diferentes métricas como: accesibilidad, rendimiento y almacenamiento, entre otros.

2.9 Condiciones iniciales para la evaluación

La velocidad de internet es necesaria para medir la métrica del comportamiento en el tiempo en las aplicaciones web tradicionales; para las PWA esta condición no es relevante, ya que por sus características propias estas mantienen su eficiencia, incluso, en redes de baja calidad. En la Tabla 6 se muestran las condiciones iniciales de velocidad de internet.

Tabla 6. Datos de MLAB de velocidad de internet. Fuente: elaboración propia.

Condiciones iniciales de la prueba de velocidad de internet	
Servidor de prueba	Bogotá, CO
Descargar	4.56 Mb/s
Subir	0.36 Mb/s
Latencia	164 ms
Retransmisión	0.91 %

Para la métrica de utilización de recursos se tomaron en cuenta las características del computador donde se ejecutaron las aplicaciones. En la Tabla 7 se observa las condiciones iniciales del computador.

Tabla 7. Condiciones iniciales del computador. Fuente: elaboración propia.

Condiciones iniciales del computador	
Sistema Operativo	Microsoft Windows 10 Home Single Languaje 64 bits
CPU	Intel Core i7-8550U – 3.7GHz
RAM	8GB – 10657 MB/s
GPU	Intel UHD Graphics 620
Disco Duro	31 (Disco Score) – (Write) 103 MB/s – (Read) 134 MB/s

La evaluación se ejecuta en 4 navegadores compatibles con trabajadores de servicio. En la Tabla 8 se observan los navegadores junto a su versión.

Tabla 8. Navegadores y su versión. Fuente: elaboración propia.

Navegadores compatibles con trabajadores de servicio	
Google Chrome	Versión 90.0.4430.93
Microsoft Edge	Versión 90.0.818.56
Opera	Versión 76.0.4017.107
Mozilla Firefox	Versión 88.0.1

La evaluación se realizó basándose en la funcionalidad más importante de la aplicación, esta es: realizar un pedido. Para las métricas de comportamiento en el tiempo se evaluaron las aplicaciones desde que el usuario ingresa a la aplicación, lista los locales, lista los productos y realiza un pedido. Para las métricas de utilización de recursos se evalúa el consumo de memoria y CPU al realizar un pedido, y para la utilización de dispositivos de E/S se toma en cuenta el tipo de almacenamiento en caché que utilizan las aplicaciones. Para las métricas de capacidad se identifica el número de peticiones procesadas y el valor máximo de transmisión necesario para hacer un pedido en la aplicación. Debido a que la PWA trabaja de forma local por cada usuario, se consideró evaluar la métrica de accesos simultáneos con un único usuario.

3. RESULTADOS Y DISCUSIÓN

Al finalizar las pruebas de rendimiento se obtuvieron resultados similares en las métricas de tiempo de respuesta y espera, debido a que la aplicación web tradicional presento otro tipo de almacenamiento similar a los trabajadores de servicio. En las métricas de utilización de

recursos, la ejecución de tareas en segundo plano ocasionó que la PWA tuviera un mayor consumo de recursos, además, no almacenó la misma cantidad de información en la caché de los navegadores. En las métricas de capacidad, las dos aplicaciones presentaron comportamientos diferentes al momento de realizar peticiones a través de la red.

Los resultados se representan en gráficas de barras: la barra azul representa a la aplicación web tradicional y la barra roja a la PWA, estos se muestran en función del navegador y el valor obtenido en cada métrica de rendimiento. A continuación, se exponen estos resultados y un análisis en cada uno de ellos.

3.1 Comportamiento en el tiempo

3.1.1 Tiempo de respuesta

En la Figura 12 se observa que a pesar de que la aplicación web tradicional no almacenó información en la caché del navegador y no instaló trabajadores de servicio, esta presenta tiempos de respuesta similares a la PWA. Esto sucede porque la aplicación web tradicional almacenó temporalmente información en la caché de disco y memoria dando como resultado una aplicación igual de eficiente. La PWA respondió a las peticiones del cliente con información almacenada en la caché del navegador, si bien aún realizaba peticiones a la caché de disco y memoria.

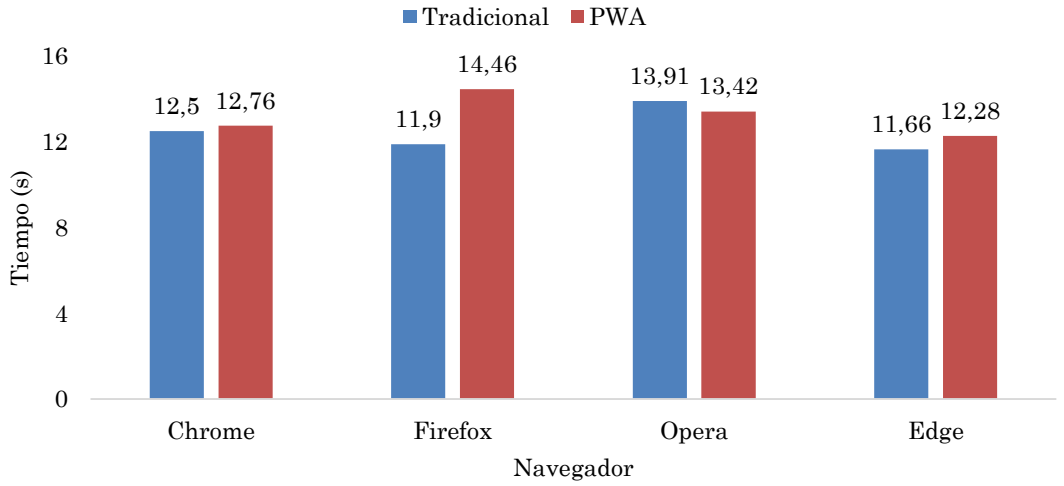

Figura 12. Tiempo de respuesta de la aplicación web tradicional vs PWA. Fuente: elaboración propia.

3.1.2 Tiempo de espera

En la Figura 13 se observa un comportamiento parecido. Por lo general, la caché de disco y memoria están habilitadas en todos los navegadores que tienen este soporte; sin embargo, al deshabilitar esta opción mediante las herramientas de desarrollador, se obtuvo como resultado tiempos de respuesta y espera más complejos en la aplicación web tradicional.

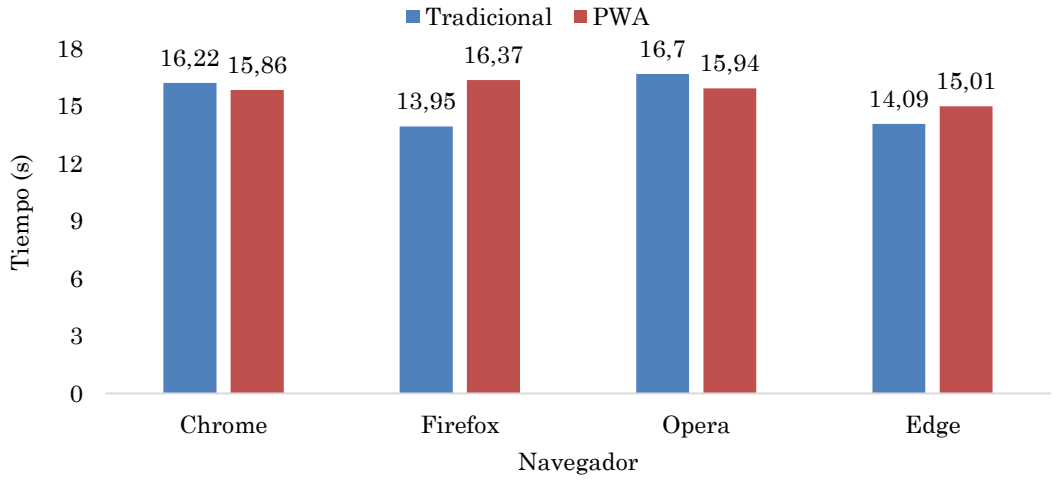

Figura 13. Tiempo de espera de la aplicación web tradicional vs PWA. Fuente: elaboración propia.

Una de las diferencias entre estos dos almacenamientos es el funcionamiento sin Internet y esto se consigue mediante el trabajador de servicio y su capacidad para guardar y responder con información almacenada en la caché del navegador.

3.1.3 Rendimiento

Para esta métrica se tomó en cuenta el número de peticiones que la aplicación hace a través de la red y el tiempo en hacerlas. En la Figura 14 se observa que la aplicación web tradicional realizó menos peticiones, en cambio, la PWA realizó más peticiones, estas provenían de la caché del disco, memoria y trabajador de servicio, por lo que su rendimiento fue mejor.

A pesar de que el almacenamiento en caché es el pilar principal para la escalabilidad web, este no debe descuidarse, ya que su uso indebido puede afectar al comportamiento de la aplicación con la pérdida de eficiencia y escalabilidad [24].

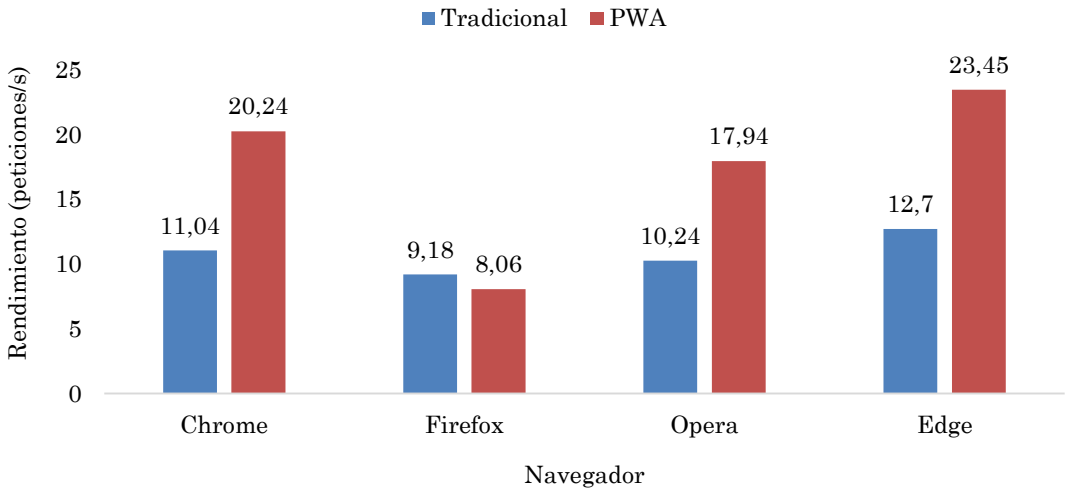

Figura 14. Rendimiento de la aplicación web tradicional vs PWA. Fuente: elaboración propia.

3.2 Utilización de recursos

3.2.1 Utilización de memoria

En la Figura 15 se observa que existió un elevado consumo de memoria en la PWA. Los trabajadores de servicio permiten a las PWA tener esa eficiencia que les caracteriza, sin embargo, cuando estos permanecían activos, el consumo de memoria fue incrementando; esto se da porque la PWA permanece ejecutando tareas en segundo plano.

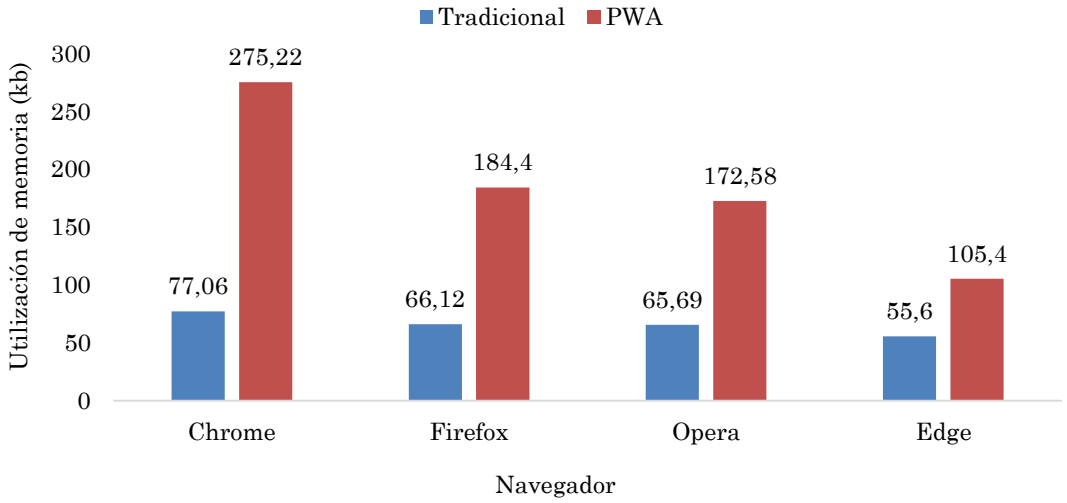

Figura 15. Utilización de memoria de la aplicación web tradicional vs PWA. Fuente: elaboración propia.

Los resultados obtenidos en la PWA se obtuvieron con tres trabajadores de servicio activos, es por eso que el consumo se duplica y en ocasiones se triplica dependiendo el navegador donde se ejecute.

3.2.2 Utilización del CPU

En la Figura 16 se observa que las aplicaciones tuvieron un comportamiento diferente con respecto al consumo del CPU, el consumo también dependió del tipo de navegador.

Para el desarrollo de aplicaciones, este tipo de consumo debe ser mínimo, ya que si el usuario sospecha que una aplicación le consume demasiados recursos terminara por desinstalarlo.

3.2.3 Utilización de dispositivos de E/S

Para esta métrica se tomó en cuenta el tiempo de espera y los tipos de almacenamiento en cada aplicación, la PWA y la aplicación web tradicional utilizaron la caché de disco y memoria, además, la PWA utilizó 3 trabajadores de servicio para el almacenamiento en la caché del navegador. En la Figura 17 se muestra el comportamiento de las aplicaciones bajo esta métrica. La PWA tuvo un mejor desempeño en esta métrica ya que contaba con más dispositivos de E/S para su almacenamiento, no obstante, presentó un comportamiento diferente con respecto al almacenamiento en caché en los distintos navegadores. En la Tabla 9 se observa el almacenamiento ocupado por esta aplicación.

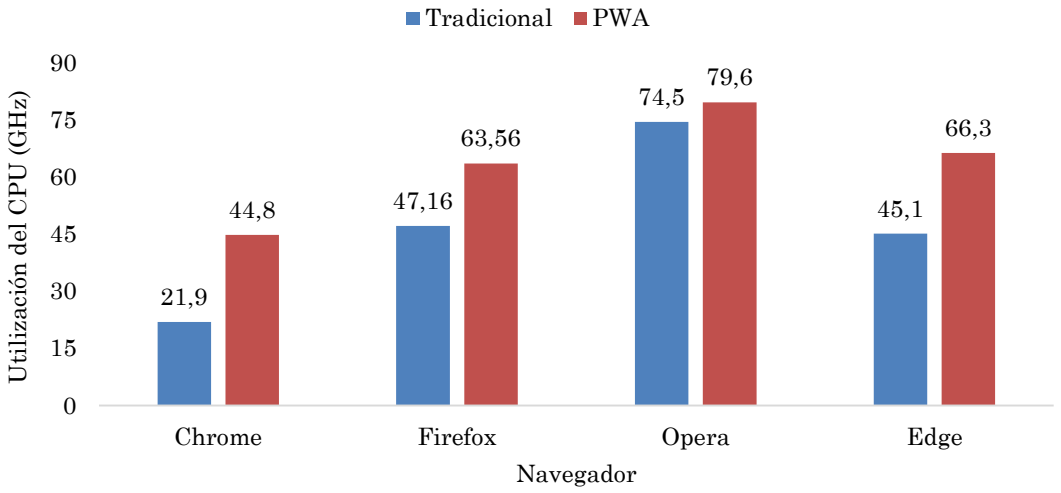

Figura 16. Utilización del CPU de la aplicación web tradicional vs PWA. Fuente: elaboración propia.

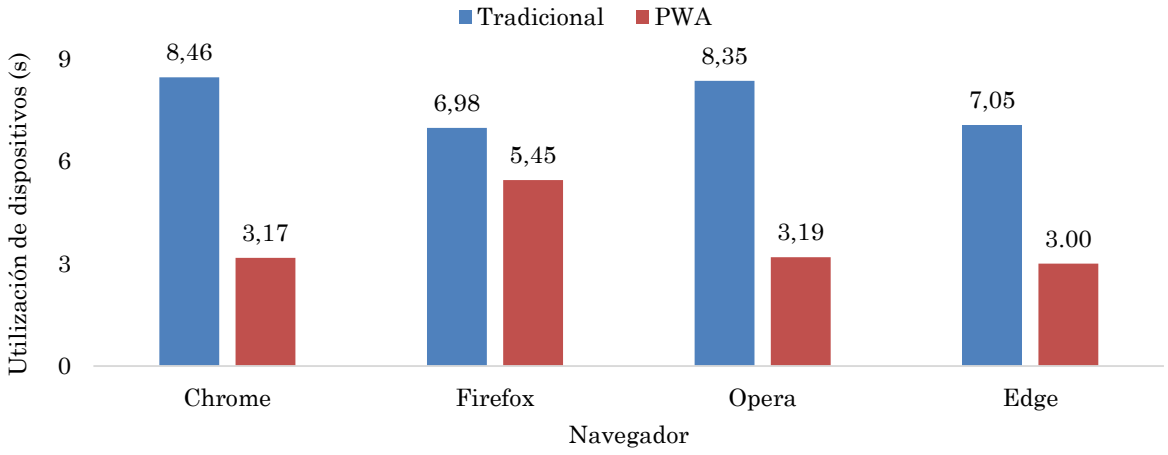

Figura 17. Utilización de dispositivos de E/S de la aplicación web tradicional vs PWA Fuente: elaboración propia.

Tabla 9. Almacenamiento en caché ocupado por la PWA. Fuente: elaboración propia.

Navegador/ Almacenamiento	Chrome	Edge	Opera	Firefox
Almacenamiento en caché	283 MB	322 MB	317 MB	52.48 MB
Trabajador de servicio	3.9 kB	3.9 kB	3.9 kB	1.8 kB
Archivos en caché	102	100	100	48

3.3 Capacidad

3.3.1 Número de peticiones en línea

Al estar las dos aplicaciones con conexión a Internet, estas realizaban peticiones al servidor para cumplir con la funcionalidad propuesta, las dos aplicaciones obtuvieron resultados similares en esta métrica, aunque, en Firefox, las peticiones de la PWA fueron mínimas, ya que únicamente realizaron peticiones a los trabajadores de servicio y al servidor para guardar el pedido. En la Figura 18 se muestra este comportamiento de las aplicaciones.

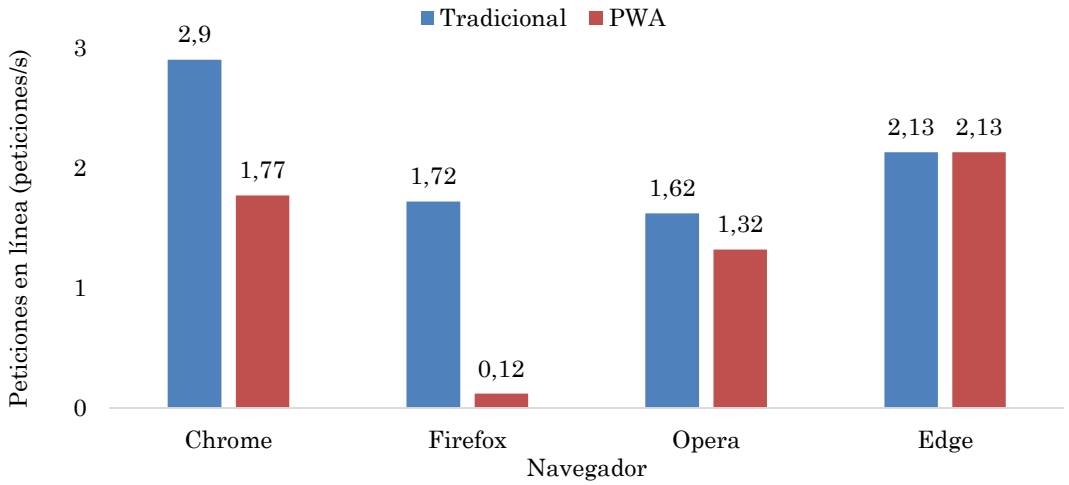

Figura 18. Número de peticiones en línea de la aplicación web tradicional vs PWA. Fuente: elaboración propia.

3.3.2 Número de accesos simultáneos

En la Figura 19 se observa que las aplicaciones presentaron resultados similares en los cuatro navegadores y las PWA trabajan de forma local por cada usuario, por lo que solo se evaluó a uno y se tomó en cuenta el tiempo de espera mencionado en las primeras métricas.

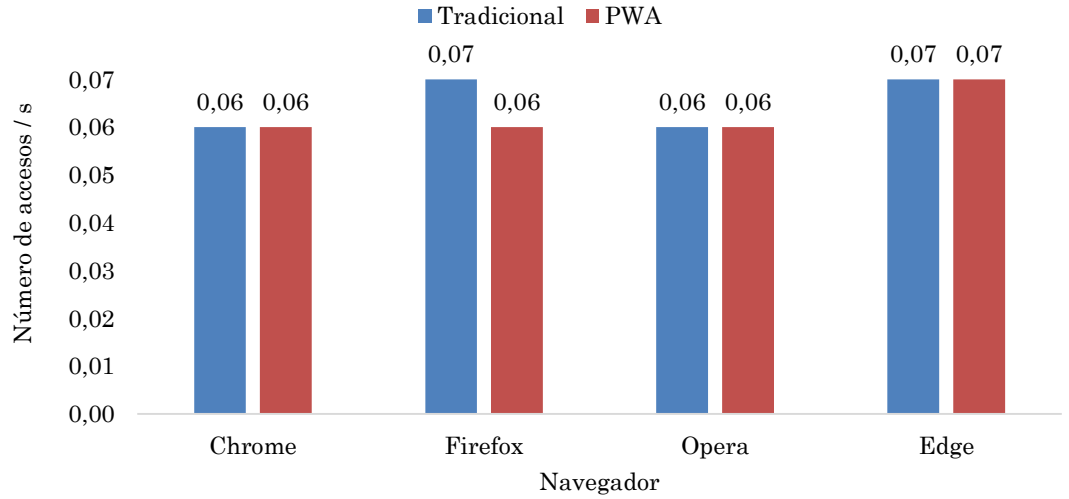

Figura 19. Número de accesos simultáneos de la aplicación web tradicional vs PWA Fuente: elaboración propia.

Probablemente los resultados aumentarían equitativamente si se evalúan las aplicaciones con más de un usuario, aun así, esto dependerá de las configuraciones que tenga la aplicación a evaluar.

3.3.3 Sistema de transmisión de ancho de banda

En la Figura 20 se observa que ambas aplicaciones transfirieron muy poca memoria a través de la red, sin embargo, el consumo de la PWA es ligeramente menor. Esto se da gracias

a la caché de disco y memoria; si no existiera este tipo de caché, las aplicaciones web tradicionales tuvieran un rendimiento sumamente bajo, en cambio, las PWA no se ven afectadas, ya que su consumo proviene de la caché del navegador.

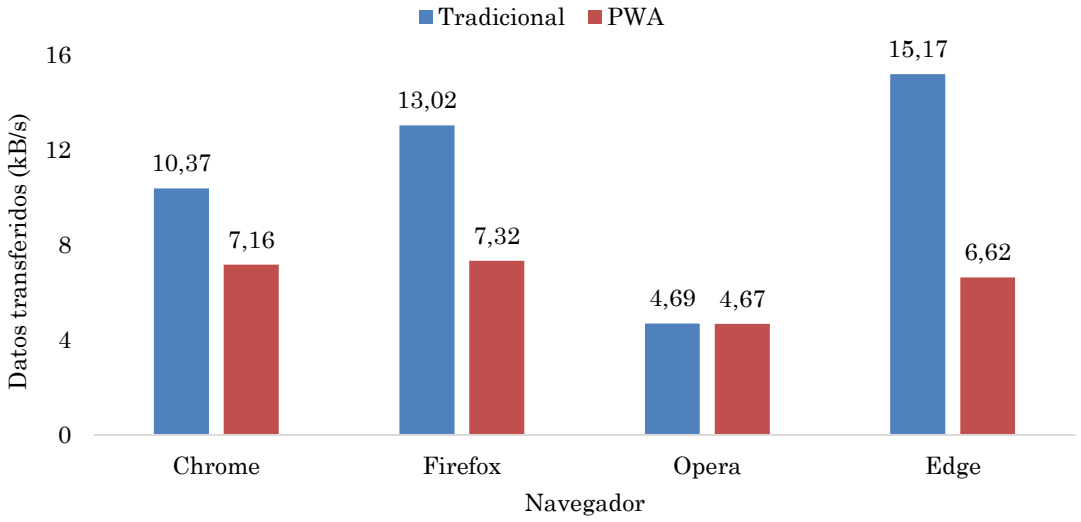

Figura 20. Cantidad de datos transferidos a través de la red de la aplicación web tradicional vs PWA
Fuente: elaboración propia.

Con estos resultados se observa que ambas aplicaciones tienen distintos comportamientos en los navegadores utilizados. Se observa la eficiencia que presenta la PWA en métricas como: comportamiento en el tiempo y capacidad; la tecnología de los trabajadores de servicio permite que una PWA tenga ciertas ventajas con respecto a las webs tradicionales, tales como: funcionamiento similar a una aplicación nativa, navegación sin Internet y uso de sensores de los dispositivos, entre otros.

Los resultados mencionados deben ser tomados en cuenta por los desarrolladores web, ya que, si las configuraciones de la PWA no son adecuadas, estas podrían afectar al comportamiento de la aplicación [24]; además, es importante considerar que cada usuario tendrá preferencias al ejecutar una aplicación web con respecto a los dispositivos y navegadores, por lo que se considera necesario poner a prueba una PWA en distintos escenarios antes de ponerla en producción.

4. CONCLUSIONES

Al finalizar la evaluación, los resultados muestran el comportamiento que tiene una PWA y una aplicación web tradicional bajo métricas de rendimiento y en navegadores compatibles con trabajadores de servicio. Estos datos permitirán a los desarrolladores web identificar la viabilidad que tiene el desarrollo con tecnología PWA y tomar precauciones con el almacenamiento en caché, ya que como se mencionó anteriormente, una mala configuración en los trabajadores de servicio puede ocasionar inestabilidad en el funcionamiento de la aplicación web.

Durante las pruebas de rendimiento se pudo verificar que una PWA no solo realizaba peticiones a los trabajadores de servicio, sino que también consultaba la caché de disco y memoria. Este consumo le dio un mejor rendimiento, si bien la ejecución de tareas en segundo

plano generó un consumo mayor de recursos del computador. Una de las características que presenta la PWA es la de simular ser una aplicación nativa para brindarle una mejor experiencia al usuario, pese a ello, no todos los navegadores tienen el mismo soporte, por lo que esto puede limitar el funcionamiento multiplataforma.

Los trabajadores de servicio son archivos fundamentales para el funcionamiento de una PWA. Para esta aplicación su función fue interceptar, almacenar y responder con información del caché, aún así se pudo observar que la PWA también almacenaba en caché información personal del usuario, por lo que una mala configuración del trabajador de servicio comprometió esta información, afectando de esa manera la seguridad de la aplicación web.

Para finalizar se concluye que el rendimiento de una PWA variará dependiendo del tipo de configuración que presenten los trabajadores de servicio y de las herramientas de *software* que se utilicen para el desarrollo web; además, su comportamiento dependerá del tipo de dispositivo y navegador en el que se ejecute la aplicación. Estas y otras métricas deben ser consideradas por los desarrolladores web para que una PWA tenga un funcionamiento eficiente y pueda reemplazar a un desarrollo nativo, siendo este último uno de los principales objetivos de esta tecnología.

5. AGRADECIMIENTOS

A la Universidad Técnica de Cotopaxi por promover la investigación científica y tecnológica. El artículo no está financiado por ninguna entidad.

CONFLICTOS DE INTERÉS

Los autores no presentan conflictos de interés.

CONTRIBUCIÓN DE LOS AUTORES

Jhonatan Llamuca-Quinaloa: desarrollo de las dos versiones de aplicaciones web, ejecución de pruebas de rendimiento, análisis de resultados y redacción del documento.

Yasmani Vera-Vincent: desarrollo de las dos versiones de aplicaciones web, ejecución de pruebas de rendimiento, análisis de resultados y redacción del documento.

Verónica Tapia-Cerda: revisión y corrección del documento.

6. REFERENCIAS

[1] O. Pinzon; K. Rodríguez, "Ingeniería Web: Una Metodología para el Desarrollo de Aplicaciones Web Escalables y Sostenibles", en *The Fifteen LACCEI International Multi-Conference for Engineering, Education Technology,* Boca Raton, Florida - US7ygthA, pp. 19–21, 2017. URL

[2] M. R. Valarezo Pardo; J. A. Honores Tapia; A. S. Gómez Moreno; L. F. Vinces Sánchez, "Comparación de tendencias tecnológicas en aplicaciones web", *3C Tecnología,* vol. 7, no. 3, pp. 28–49, 2018. http://dx.doi.org/10.17993/3ctecno.2018.v7n3e27.28-49/

[3] F. Shahzad, "Modern and Responsive Mobile-enabled Web Applications", *Procedia Comput. Sci.,* vol. 110, pp. 410–415, 2017. http://dx.doi.org/10.1016/j.procs.2017.06.105

[4] "Live Stats", 2020. URL

[5] J. I. Xool-Clavel; H. F. Buenfil-Paredes; M. E. Dzul-Canche, "Desarrollo e implementación de un sistema web para el proceso de estadía", *Revista de Tecnologías de la Información y Comunicaciones*, vol. 2, no. 3,

pp. 8–19, Mar. 2018. URL
[6] V. Aguirre *et al.,* "PWA para unificar el desarrollo Desktop, Web y Mobile", en *XXV Congreso Argentino de Ciencias de la Computación (CACIC)*, Córdoba, 2019, pp. 778–786. URL
[7] Mozilla and individual contributors, "Introducción a aplicaciones web progresivas", 2020. URL
[8] P. Loreto; J. Braga; H. Peixoto; J. Machado; A. Abelha, "Step towards progressive web development in obstetrics", *Procedia Comput. Sci*., vol. 141, pp. 525–530, 2018. https://doi.org/10.1016/j.procs.2018.10.131
[9] P. A. Roa; C. Morales; P. Gutiérrez, "Norma ISO/IEC 25000", Tecnol. Investig. y Acad., vol. 3, no. 2, pp. 26–32, Dec. 2015. URL
[10] ISO/IEC 25023, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality", 2016. URL
[11] ISO/IEC 25010, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models", 2011. URL
[12] ISO/IEC 25000, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE", 2014. URL
[13] L. G. Morrison *et al.,* "Comparing usage of a web and app stress management intervention: An observational study", *Internet Interv.,* vol. 12, pp. 74–82, Jun. 2018. https://doi.org/10.1016/j.invent.2018.03.006
[14] S. Tandel; A. Jamadar, "Impact of Progressive Web Apps on Web App Development", *Int. J. Innov. Res. Sci. Eng. Technol.,* vol. 7, no. 9, pp. 9439–9444, Sep. 2018. URL
[15] A. Biørn-Hansen; T. A. Majchrzak; T. M. Grønli, "Progressive web apps: The possible web-native unifier for mobile development", en *Proceedings of the 13th International Conference on Web Information Systems and Technologies*, Porto, 2017, pp. 344–351. http://dx.doi.org/10.5220/0006353703440351
[16] T. A. Majchrzak; A. Biørn-Hansen; T.-M. Grønli, "Progressive Web Apps: the Definite Approach to Cross-Platform Development?", en *Proceedings of the 51st Hawaii International Conference on System Sciences*, Hawaii, 2018, pp. 5735–5744. http://dx.doi.org/10.24251/HICSS.2018.718
[17] ISO 25000, "La familia de normas ISO/IEC 25000", s.f. URL
[18] A. J. Durán Sanjuán; J. L. Peinado Rodríguez; A. A. Rosado Gomez, "Comparación de dos tecnologías de desarrollo de aplicaciones móviles desde la perspectiva de los atributos de calidad", *Sci. Tech*., vol. 20, no. 1, pp. 81–87, Mar. 2015. https://doi.org/10.22517/23447214.9278
[19] E. F. Maila Maila, "Evaluación de herramientas Open Source para pruebas de fiabilidad y rendimiento de aplicaciones web", (Tesis de grado), Facultad de Ingeniería de Sistemas, Escuela Politécnica Nacional, 2018. URL
[20] K. G. Rodriguez; O. J. Ortiz; A. I. Quiroz; M. L. Parrales, "El e-commerce y las Mipymes en tiempos de Covid-19", *Espacios*, vol. 41, no. 42, pp. 100–118, 2020. https://doi.org/10.48082/espacios-a20v41n42p09
[21] D. Palacios; J. Guamán; S. Contento, "Análisis del rendimiento de librerías de componentes Java Server Faces en el desarrollo de aplicaciones web", *Novasinergia Rev. Digit. Ciencia, Ing. Y Tecnol.,* vol. 1, no. 2, pp. 54–59, 2018. https://doi.org/10.37135/unach.ns.001.02.06
[22] J. Keith, "HTTPS + service worker + web app manifest = progressive web app", 2018. URL
[23] M. Santoni, "Las Progressive Web Apps: compatibilidad de las funcionalidades en función de los navegadores", GoodBarber, 2018. URL
[24] H. V. Nguyen; L. Lo Iacono; H. Federrath, "Systematic Analysis of Web Browser Caches", *WS.2 2018: Proceedings of the 2nd International Conference on Web Studies,* Paris, 2018, pp. 64–71. https://doi.org/10.1145/3240431.3240443
[25] Measurement Lab, "Test Your Speed", 2021. URL
[26] Novabench, "Your computer at its best", 2007. URL
[27] A. Vykhodtsev, "Page load time", 2021. URL