JURNAL ILMU KOMPUTER
KHARISMA .TECH

Jurnal KHARISMA Tech | ISSN:1907-2317 | e-ISSN: 2810-0344
https://tech.kharisma.ac.id
published by: Pusat Penelitian STMIK KHARISMA Makassar

Volume: 18 no. 02 – September 2023- hlm. 110-124

IMPLEMENTASI PROGRESSIVE WEB APPLICATION PADA *WEBSITE* FRIZFOO MENGGUNAKAN EXPRESS.JS

Oleh:

Theodorus Karli[1], Ahyar Muawwal[2*], Mohammad Sofyan S. Thayf[3]
[1,2,3]Sistem Informasi, STMIK KHARISMA Makassar
e-mail: [1]theodoruskarli_20@kharisma.ac.id, [2]ahyar@kharisma.ac.id,
[3]sofyan.thayf@kharisma.ac.id

Abstrak: *Penelitian ini bertujuan untuk mengimplementasi teknologi Progressive Web Apps (PWA) pada website Frizfoo menggunakan framework Express.js sehingga dapat diakses dalam bentuk aplikasi native melalui home screen smartphone pengguna dan dalam keadaan offline. Penerapan PWA sebagai salah satu fitur "should have" bertujuan untuk memberikan nilai tambah dan meningkatkan performance website, serta mengatasi keterbatasan website dalam menampilkan halaman offline, dan menghindari biaya pengembangan aplikasi native di berbagai platform sistem operasi. Metode pengumpulan data melibatkan studi literatur dan pengukuran langsung menggunakan beberapa tools. Pengujian yang dilakukan meliputi, pengujian installable, kriteria PWA, performance, size transferred resources, offline mode. Komponen yang digunakan dalam PWA meliputi web app manifest, service worker, cache storage. Implementasi PWA melibatkan pembuatan web app manifest, file registrasi service worker, konfigurasi file service worker, penambahan tag script, pembuatan route khusus dalam Express.js, pengujian PWA. Hasil pengujian menunjukkan website Frizfoo dapat diinstal dan digunakan dengan baik pada berbagai tipe perangkat mobile. PWA berhasil diimplementasikan dan memenuhi seluruh kriteria PWA, dengan mendapatkan skor total 10 pada Lighthouse dan skor total 25 pada PWABuilder. Hasil pengujian performance pada Lighthouse dan GTMetrix mengalami peningkatan sebesar 36% setelah implementasi PWA. Ukuran size transferred resources berkurang dari 3.1 MB menjadi 1.8 MB dan waktu load website berkurang dari 3,65 detik menjadi 826 ms, sehingga website dapat dimuat lebih cepat. Dan Fitur offline mode website Frizfoo berhasil digunakan dalam kondisi offline atau koneksi internet yang tidak stabil.*

Kata kunci: *Progressive Web Apps, PWA, Express.js, Frizfoo, Makanan Beku*

Abstract: *The aim of this study is to implement Progressive Web Apps (PWA) technology on the Frizfoo website using the Express.js framework so that it can be accessed in the form of native applications via the user's smartphone home screen and can be run offline. The implementation of PWA as one of the "should have" features aims to provide added value and improve website performance, as well as overcomes website limitations in displaying offline pages, and avoid the cost of developing native applications on various operating systems platforms. Data collection methods involve literature study and direct measurement using several tools. Testing included installable testing, PWA criteria, performance, size transferred resources, and offline mode. Components used in PWA include web app manifest, service worker, and cache storage. PWA implementation involves creating a web app manifest, service worker registration file, configuring service worker files, adding script tags, creating custom routes in Express.js, and testing PWA. The test results show that the Frizfoo website can be installed and used properly on various types of mobile devices. PWA was successfully implemented and met all PWA criteria, by getting a total score of 10 on Lighthouse and a total score of 25 on PWABuilder. The results of performance testing on Lighthouse and GTMetrix have increased by 36% after the implementation of PWA. The size of transferred resources*

[*] Corresponding author : Ahyar Muawwal (ahyar@kharisma.ac.id)

Diterima : *September, 2023* ***Disetujui :*** *September, 2023* ***Dipublikasikan :*** *Oktober, 2023*

has reduced from 3.1 MB to 1.8 MB and the website load time has decreased from 3.65 seconds to 826 ms so that the website can be loaded faster. And offline mode feature of the Frizfoo website is successfully used in offline conditions or with an unstable internet connection.

Keywords: *Progressive Web Apps, PWA, Express.js, Frizfoo, Frozen Food*

1. PENDAHULUAN

Frizfoo yang saat ini dapat diakses melalui alamat https://frizfoo.com/ merupakan sebuah platform berupa *web application* yang dapat menghubungkan pembeli dan penjual dalam satu komunitas, tetapi hanya berfokus pada produk *frozen food*. *Website* Frizfoo dibangun dengan konsep *Minimum Viable Product* (MVP), yang diperkenalkan oleh Eric Ries dalam bukunya yang berjudul "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses". Konsep MVP berfokus pada pendekatan fitur dasar yang cukup untuk memenuhi kebutuhan dasar pengguna, sehingga pengembang dapat meluncurkan produk mereka dengan cepat tanpa mengejar kesempurnaan [1]. Dalam pengembangan produk berbasis MVP, terdapat istilah "*must have*" dan "*should have*" untuk mengatur prioritas fitur dan fungsi produk. Sehingga, peneliti melakukan implementasi PWA pada *website* Frizfoo untuk melanjutkan penambahan fitur "*should have*" *website* Frizfoo, serta untuk memberikan nilai tambah dan meningkatkan kualitas dari *website* Frizfoo.

Aplikasi *Native* merupakan aplikasi yang dikembangkan khusus untuk digunakan pada satu platform sistem operasi, seperti Android atau iOS. Sehingga hanya dapat berjalan di salah satu sistem operasi dan tidak dapat digunakan secara lintas platform [2]. Oleh karena itu, peneliti ingin melakukan implementasi PWA pada *website* Frizfoo untuk mengatasi beberapa masalah seperti dalam hal biaya pengembangan dan perawatan jika ingin membuat aplikasi *native* Frizfoo di berbagai platform sistem operasi, meningkatkan *performance website* Frizfoo khususnya dalam hal kecepatan *loading* halaman *website* dikarenakan hasil pengujian menggunakan Google Lighthouse menunjukkan perlunya penerapan *cache policy*, serta mengatasi masalah keterbatasan *website* Frizfoo agar dapat menampilkan halaman statis khusus saat *user* sedang *offline*.

Tujuan dari studi ini yaitu untuk mengimplementasi teknologi Progressive Web Apps (PWA) pada *website* Frizfoo menggunakan Express.js agar bisa digunakan dalam bentuk aplikasi *native* pada perangkat *smartphone* melalui *home screen smartphone*, tanpa harus menginstal terlebih dahulu melalui Play Store atau App Store, serta dapat digunakan dalam keadaan *user* sedang *offline*. PWA merupakan teknologi *Web-based development*, yang memungkinkan agar suatu *website* dapat memiliki karakteristik / *experience* layaknya menggunakan suatu aplikasi *mobile native*, sehingga dapat memberikan pengalaman yang berkesan kepada *user* [3]. Teknologi PWA dikembangkan oleh Google dan para pengembang *browser* dan *mobile* lainnya untuk mempermudah dalam pembuatan aplikasi *multi-platform* [4]. Dengan menggunakan *service worker*, *web app manifest*, serta *cache API*, PWA memiliki

beberapa fitur seperti *offline mode*, instalasi pada *home screen* (*Add To Home Screen*), *push notification*, *background sync*, dan *splash screen* [5].

Express.js merupakan *web app framework* yang dibangun diatas NodeJs (*Javascript Runtime Environment*) yang termasuk kategori *framework* paling populer di dunia Node.Js karena terkenal sebagai *framework* yang minimalis serta fleksibel jika ingin membuat sebuah aplikasi web [6]. Adapun pertimbangan implementasi PWA pada *website* Frizfoo menggunakan Express.js, yaitu seperti kemudahan dalam mengelola *template* statis *website* Frizfoo karena terintegrasi dengan berbagai *template engine*, proses *rendering website* di sisi *server* (*server side rendering*) sehingga halaman *website* dapat ditampilkan lebih cepat, memastikan *SEO friendly* dan *performance website* yang baik, kemudahan dalam *setup* dan *deploy project*, memiliki fitur pengembangan API dan dukungan *third-party middleware*, bersifat *unopinionated* yang mana fleksibel dalam hal kostumisasi struktur folder aplikasi dan fungsi *middleware*.

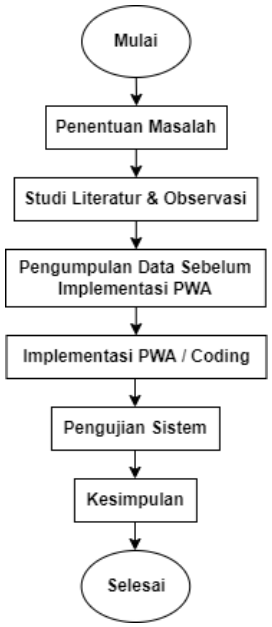
Penelitian terkait implementasi teknologi Progressive Web Apps pada suatu *website* telah pernah dilakukan oleh beberapa peneliti sebelumnya. Seperti dalam penelitian [7] yang dilakukan oleh Aripin, Somantri berjudul “Implementasi Progressive Web Apps (PWA) pada Repository E-Portofolio Mahasiswa”. Penelitian [8] yang dilakukan oleh Riady, Palit, Andjarwirawan dengan judul “Aplikasi E-Learning Berbasis Progressive Web App Pada Apologetika Indonesia”. Sedangkan, penelitian yang dilakukan oleh Bahari, Sumaryana [9] dengan judul “Penerapan Progressive Web Apps Pada Aplikasi Lowongan Pekerjaan Dosen Universitas Perjuangan”. Dan juga, pada penelitian [10] yang dilakukan oleh Haryanto, Elsi dengan judul “Analisis Performance Progressive Web Apps Pada Aplikasi Shopee”. Selain itu, penelitian [11] yang dilakukan oleh Aslan, Bahtiar, Sudianto dengan judul “Pengembangan Website Fakultas Teknik Universitas Hamzanwadi Berbasis Progressive WEB APP (PWA)”. Demikian pula, penelitian yang dilakukan oleh Joarno, Fajar, Yunus [12] dengan judul “Implementasi Progressive Web Apps Pada Website Gethelp Menggunakan Next.js”. Dalam studi ini, digunakan *framework* Express.js yang berbeda dengan penelitian sebelumnya. Selain itu, dilakukan pengujian *installable* setelah implementasi PWA, terpenuhinya suatu kriteria PWA, *performance website*, *size transferred resources*, dan *offline mode* baik dalam keadaan sebelum dan sesudah implementasi PWA.

2. METODE PENELITIAN

2.1 Tahapan Penelitian

Pada tahap pengujian sistem, berbagai pengujian dilakukan seperti pengujian *installable* menggunakan berbagai macam tipe perangkat *mobile* dan *browser*, pengujian terpenuhinya suatu kriteria PWA menggunakan Lighthouse terdiri dari dua parameter (*Installable*, *PWA Optimized*) dan pengujian manual, sedangkan PWABuilder terdiri dari tiga parameter (*Service worker*, *Manifest*, *Security*) dengan masing-masing parameter terdapat beberapa kriteria yang diuji. Pengujian *performance* menggunakan Lighthouse berfokus pada enam variabel pengujian dan GTMetrix berfokus pada 12 variabel pengujian. Dan untuk pengujian *size transferred resources*, dan *offline mode website* Frizfoo menggunakan *browser DevTools*.

Sehingga hasil pengujian diolah menjadi tabel untuk dibandingkan dan dianalisis antara sebelum dan sesudah implementasi PWA, sehingga dapat ditarik beberapa kesimpulan terkait penelitian yang dilakukan. Untuk tahapan penelitian, disajikan pada Gambar 1 di bawah ini.



Gambar 1. Tahapan Penelitian

2.2 Jenis Data dan Sumber Data

Jenis data yang digunakan dalam penelitian ini yaitu data kuantitatif, yang diperoleh dari hasil eksperimen berupa pengukuran langsung *website* Frizfoo dengan menggunakan beberapa *tools* seperti *Browser* Edge dan Chrome, PWABuilder, Google Lighthouse, GTMetrix, Chrome *DevTools*. Data tersebut mencakup pengujian *installable* setelah implementasi PWA, terpenuhinya suatu kriteria PWA, *performance website*, *size transferred resources*, dan *offline mode* dalam keadaan sebelum dan sesudah implementasi PWA. Selain data primer tersebut, sumber data sekunder diperoleh dari hasil studi pustaka atau *library research* berupa artikel jurnal yang relevan dengan penelitian terkait arsitektur pengimplementasian PWA, komponen dan fitur utama PWA seperti *offline mode*, *add to home screen*, penggunaan *cache storage*, *service worker life cycle*, dan lain sebagainya.

3. HASIL DAN PEMBAHASAN

3.1 Implementasi PWA / Coding

Terdapat beberapa tahapan dalam melakukan implementasi PWA pada *website* Frizfoo menggunakan Express.js, yaitu:

1. Penambahan *Web App Manifest*

Web App Manifest merupakan sebuah *file* JSON (*Javascript Object Notation*) yang memberikan informasi detail kepada *browser* tentang aplikasi web dan bagaimana perilakunya ketika diinstal [13]. Pada tahap ini, penulis membuat sebuah *Web App Manifest* yang diberi nama manifest.json, untuk menyimpan seluruh informasi detail *website* Frizfoo, seperti nama

aplikasi, ikon, deskripsi, warna tema, orientasi layar, dan lain sebagainya, sehingga *website* Frizfoo dapat memiliki fitur *add to home screen* dan *splash screen* saat aplikasi web dibuka. *File* manifest.json dapat dilihat pada Gambar 2 di bawah ini.

```
{} manifest.json > ...
{
    "name": "Frizfoo",
    "short_name": "Frizfoo",
    "description": "Frizfoo merupakan Suatu Web Application yang dapat menghubungkan pembeli dan penjual dalam satu komunitas, tetapi hanya berfokus pada produk Frozen Food.",
    "lang": "id",
    "dir": "ltr",
    "icons": [
        {
            "src": "/img/pwaicons/frizfoo-icons-192.png",
            "sizes": "192x192",
            "type": "image/png",
            "purpose": "any"
        },
        {
            "src": "/img/pwaicons/frizfoo-icons-512.png",
            "sizes": "512x512",
            "type": "image/png",
            "purpose": "any"
        },
        {
            "src": "/img/pwaicons/frizfoo-maskable-icons-682.png",
            "sizes": "682x682",
            "type": "image/png",
            "purpose": "maskable"
        }
    ],
    "start_url": "/",
    "id": "/",
    "background_color": "#FFFFFF",
    "display": "standalone",
    "display_override": [
        "window-controls-overlay"
    ],
    "edge_side_panel": {
        "preferred_width": 400
    },
    "categories": [
        "food",
        "business",
        "e-commerce",
        "community",
        "marketplace"
    ],
    "scope": "/",
    "theme_color": "#c1ccfd",
```

Gambar 2. *File* manifest.json

Setelah membuat *file* manifest.json, langkah selanjutnya yaitu menambahkan *tag link* pada *header* halaman *website* Frizfoo. Hal ini dilakukan untuk memuat *file* manifest.json dan memberitahu *browser* bahwa *website* Frizfoo mengadopsi teknologi PWA dan bersifat *installable*, seperti yang dapat dilihat pada Gambar 3 dibawah ini.

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

Gambar 3. Pemanggilan manifest.json Pada *Header Website* Frizfoo

Selain itu, penulis perlu membuat *route* khusus untuk menangani permintaan /manifest.json, dikarenakan *file* manifest.json ditempatkan di *root directory* pada folder aplikasi *website* Frizfoo dan bukan pada folder statis (public) yang telah ditentukan menggunakan *built-in middleware* bawaan Express.js. Sehingga *route* untuk menangani permintaan /manifest.json, dapat dilihat pada Gambar 4 di bawah ini.

```
app.get('/manifest.json', (req, res) => {
    res.sendFile(path.join(__dirname, 'manifest.json'));
});
```

Gambar 4. *Request Route* manifest.json

2. Penambahan *Service Worker*

Service Worker merupakan *script* Javascript yang berjalan di latar belakang *browser* pengguna dan menjadi pintu masuk dari fitur PWA seperti *push notifications*, *background sync*, *caching resources*, *offline mode*, dan lain sebagainya [14]. Pada tahap ini, penulis membuat sebuah *file* bernama swregist.js untuk konfigurasi registrasi *service worker*. Selanjutnya, dilakukan penambahan *tag script* agar *file* swregist.js dapat dimuat, dan *service worker* dapat diregistrasikan pada *website* Frizfoo menggunakan *file* yang ada pada *route* /sw.js, seperti yang dapat dilihat pada Gambar 5 dan Gambar 6 di bawah ini.

```
if ('serviceWorker' in navigator) {
    window.addEventListener('load', function () {
        navigator.serviceWorker.register('/sw.js').then(function (registration) {
            console.log('ServiceWorker registration successful with scope: ',
            registration.scope);
        }, function (err) {
            console.log('ServiceWorker registration failed: ', err);
        });
    });
}
```

Gambar 5. Konfigurasi Registrasi *Service Worker*

```
<script src="/js/swregist.js"></script>
```

Gambar 6. Pemanggilan *Script* swregist Untuk Registrasi *Service Worker*

```
var CACHE_NAME = 'frizfoo-cache-v1';
var urlsToCache = [
    '/',
    '/offline',
    '/css/style.css',
    '/js/navbarscript.js',
    '/js/scrolltotop.js',
    '/js/subscribeform.js',
    '/img/logo.webp',
    '/img/offline.gif',
];

self.addEventListener('install', function (event) {
    event.waitUntil(
        caches.open(CACHE_NAME)
            .then(function (cache) {
                console.log('Opened cache');
                return cache.addAll(urlsToCache);
            })
    );
});

self.addEventListener('fetch', function (event) {
    event.respondWith(
        caches.match(event.request)
            .then(function (response) {
                if (response) {
                    return response;
                }

                return fetch(event.request).then(
                    function (response) {
                        if (!response || response.status !== 200 || response.type !== 'basic') {
                            return response;
                        }
                        var responseToCache = response.clone();
                        caches.open(CACHE_NAME)
                            .then(function (cache) {
                                cache.put(event.request, responseToCache);
                            });
                        return response;
                    }
                )
            }).catch(function () {
                return caches.match('/offline');
            })
    );
});

self.addEventListener('activate', function (event) {
    var cacheAllowlist = CACHE_NAME;
    event.waitUntil(
```

Gambar 7. *File* Konfigurasi *Service Worker*

```
app.get('/sw.js', (req, res) => {
    res.sendFile(path.join(__dirname, 'sw.js'));
});
```

Gambar 8. *Request Route* sw.js

Selanjutnya, penulis membuat sebuah *file* baru bernama sw.js di dalam *root directory* yang berisikan seluruh konfigurasi *service worker* untuk *website* Frizfoo. Setelah mengkonfigurasi *file* sw.js, penulis membuat *route* khusus untuk menangani permintaan /sw.js, dikarenakan *file* sw.js ditempatkan di *root directory* dan bukan pada folder statis (public) yang telah ditentukan oleh Express.js. Detail *file* konfigurasi *service worker* dan *route* untuk menangani permintaan /sw.js dapat dilihat pada Gambar 7 dan Gambar 8 di atas.

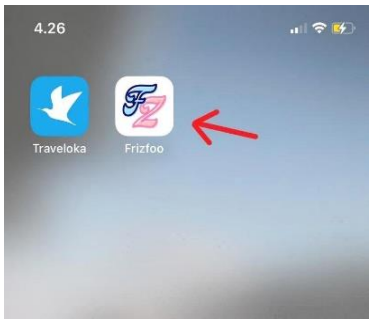
3.2 Pengujian Sistem

Terdapat beberapa macam pengujian sistem yang dilakukan untuk memastikan keberhasilan dalam implementasi PWA pada *website* Frizfoo, yaitu:

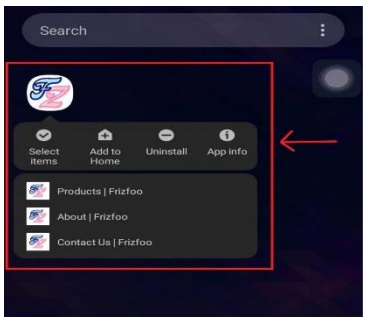
1. Pengujian *Installable*

Pengujian *Installable* dilakukan pada perangkat *mobile* dan *desktop*, untuk memastikan bahwa *website* Frizfoo dapat diinstal pada berbagai perangkat pengguna. Pengujian *desktop installable* dilakukan menggunakan *browser* Google Chrome dan Microsoft Edge, sedangkan pengujian *mobile installable* dilakukan pada lima perangkat yang berbeda, termasuk iPhone XS, Xiaomi Redmi 10, Samsung Galaxy A6, Samsung Galaxy A30, dan Vivo V20.

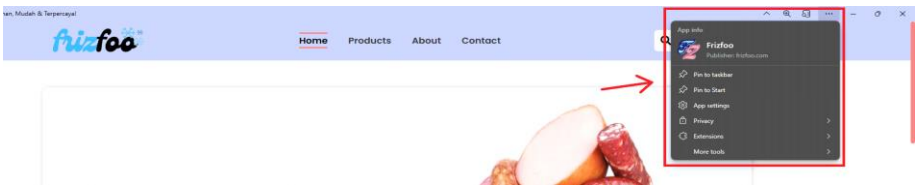
Pengujian *mobile installable* dan *desktop installable* dilakukan pada perangkat dengan platform yang berbeda, agar dapat mengamati kemampuan instalasi *website* Frizfoo yang telah mengadopsi teknologi PWA. Dan didapatkan bahwa *website* Frizfoo telah berhasil diinstal pada berbagai perangkat *mobile* dan *desktop* pengguna, sehingga diberikan nilai 1 sebagai indikasi keberhasilan *website* Frizfoo dalam pengujian *mobile installable* dan *desktop installable*, seperti yang dapat dilihat pada Gambar 9 – Gambar 11 di bawah ini.



Gambar 9. *Icon* Aplikasi di *Home Screen* iPhone XS



Gambar 10. *Icon* Aplikasi di *Home Screen* Samsung Galaxy A6



Gambar 11. Aplikasi *Desktop* Frizfoo Microsoft Edge

2. Pengujian Kriteria PWA

a. Lighthouse

Hasil pengujian terpenuhinya kriteria PWA *website* Frizfoo menggunakan Lighthouse, baik dalam keadaan sebelum dan sesudah implementasi PWA, yaitu:

Tabel 1: Hasil Pengujian Kriteria PWA Menggunakan Lighthouse

[Sumber : Hasil Pengujian Kriteria PWA *Website* Frizfoo Menggunakan Lighthouse, 2023]

No.	Kriteria	Sebelum Implementasi PWA		Setelah Implementasi PWA	
		Status	Nilai	Status	Nilai
Installable					
1.	*Web app manifest* atau *service worker* memenuhi persyaratan instalasi aplikasi	Tidak Terpenuhi	0	Terpenuhi	1
PWA Optimized					
2.	Mendaftarkan *service worker* yang mengontrol halaman dan *start_url*	Tidak Terpenuhi	0	Terpenuhi	1
3.	Konfigurasi untuk *splash screen* kustom	Tidak Terpenuhi	0	Terpenuhi	1
4.	Mengatur warna tema untuk *address bar*	Tidak Terpenuhi	0	Terpenuhi	1
5.	Ukuran konten tepat untuk *viewport*	Terpenuhi	1	Terpenuhi	1
6.	Mempunyai *tag* <meta name=”viewport”> dengan atribut *width* atau *initial-scale*	Terpenuhi	1	Terpenuhi	1
7.	*Manifest* memiliki *maskable icon*	Tidak Terpenuhi	0	Terpenuhi	1
Manually Check					
8.	Situs dapat dijalankan di berbagai *browser* (*cross-browser*)	Tidak Terpenuhi	0	Terpenuhi	1
9.	Transisi halaman terasa cepat dan tidak terasa seperti diblokir di jaringan	Tidak Terpenuhi	0	Terpenuhi	1
10.	Setiap halaman memiliki *URL*	Tidak Terpenuhi	0	Terpenuhi	1
Total Nilai			2		10

Berdasarkan Tabel 1, terlihat bahwa skor yang didapatkan sebelum *website* diimplementasikan PWA yaitu sebesar 2 dimana hanya 2 kriteria pengujian yang terpenuhi. Sedangkan setelah penulis berhasil melakukan pengimplementasian PWA, maka didapatkan

skor 10 dengan 10 kriteria pengujian yang terpenuhi. Sehingga hasil pengujian tersebut menunjukkan bahwa teknologi PWA berhasil diimplementasikan dan sudah memenuhi seluruh kriteria yang telah direkomendasikan oleh Lighthouse.

b. PWABuilder

Hasil pengujian terpenuhinya kriteria PWA *website* Frizfoo menggunakan PWABuilder, baik dalam keadaan sebelum dan sesudah implementasi PWA, yaitu:

Tabel 2: Hasil Pengujian Kriteria PWA Menggunakan PWABuilder

[Sumber : Hasil Pengujian Kriteria PWA *Website* Frizfoo Menggunakan PWABuilder, 2023]

No.	Kriteria	Sebelum Implementasi PWA		Setelah Implementasi PWA	
		Status	Nilai	Status	Nilai
1.	*Manifest* memiliki properti *icon*	Tidak Terpenuhi	0	Terpenuhi	1
2.	*Manifest* memiliki properti *name*	Tidak Terpenuhi	0	Terpenuhi	1
3.	*Manifest* memiliki properti *short_name*	Tidak Terpenuhi	0	Terpenuhi	1
4.	*Manifest* memiliki properti *start_url*	Tidak Terpenuhi	0	Terpenuhi	1
5.	*Manifest* memiliki properti *background_color* dalam format *hex*.	Tidak Terpenuhi	0	Terpenuhi	1
6.	*Manifest* memiliki properti *description*	Tidak Terpenuhi	0	Terpenuhi	1
7.	*Manifest* memiliki properti *display*	Tidak Terpenuhi	0	Terpenuhi	1
8.	*Manifest* memiliki properti *display_override*	Tidak Terpenuhi	0	Terpenuhi	1
9.	*Manifest* memiliki properti *edge_side_panel*	Tidak Terpenuhi	0	Terpenuhi	1
10.	*Icons* memiliki setidaknya satu *icon* dengan tujuan apapun	Tidak Terpenuhi	0	Terpenuhi	1
11.	*Manifest* memiliki ID aplikasi	Tidak Terpenuhi	0	Terpenuhi	1
12.	*Manifest* memiliki properti *orientation*	Tidak Terpenuhi	0	Terpenuhi	1
13.	*Manifest* memiliki properti *screenshots*	Tidak Terpenuhi	0	Terpenuhi	1
14.	*Manifest* memiliki properti *theme_color* dalam format *hex*.	Tidak Terpenuhi	0	Terpenuhi	1
15.	*Manifest* memiliki properti *categories*	Tidak Terpenuhi	0	Terpenuhi	1
16.	*Manifest* menentukan arah teks *default*	Tidak Terpenuhi	0	Terpenuhi	1
17.	*Manifest* menentukan bahasa yang digunakan	Tidak Terpenuhi	0	Terpenuhi	1

18.	*Manifest* mengatur properti *prefer_related_applications* dengan benar	Tidak Terpenuhi	0	Terpenuhi	1
19.	*Manifest* memiliki properti *related_applications*	Tidak Terpenuhi	0	Terpenuhi	1
20.	*Manifest* memiliki properti *scope*	Tidak Terpenuhi	0	Terpenuhi	1
21.	*Manifest* memiliki properti *shortcuts*	Tidak Terpenuhi	0	Terpenuhi	1
22.	Memiliki *service worker*	Tidak Terpenuhi	0	Terpenuhi	1
23.	Menggunakan HTTPS	Terpenuhi	1	Terpenuhi	1
24.	Memiliki sertifikat SSL yang valid	Terpenuhi	1	Terpenuhi	1
25.	Tidak ada konten campuran di halaman	Terpenuhi	1	Terpenuhi	1
Total Nilai			3		25

Berdasarkan Tabel 2, terlihat bahwa skor yang didapatkan sebelum *website* diimplementasikan PWA yaitu sebesar 3 dimana hanya 3 kriteria pengujian yang terpenuhi. Sedangkan setelah penulis berhasil melakukan implementasi PWA, maka didapatkan skor 25 dengan 25 kriteria pengujian yang terpenuhi. Sehingga hasil pengujian tersebut menunjukkan bahwa teknologi PWA telah berhasil diimplementasikan dan bahkan melebihi beberapa kriteria yang telah direkomendasikan oleh PWABuilder, seperti *Service Worker* (satu kriteria *Highly Recommended*), *Manifest* (empat kriteria *Required*), *Security* (tiga kriteria *Required*) yang harus terpenuhi agar suatu *website* dapat dikatakan mengadopsi teknologi PWA.

3. Pengujian *Performance*

a. Lighthouse

Hasil pengujian *performance website* Frizfoo, baik dalam keadaan sebelum dan sesudah implementasi PWA menggunakan Lighthouse, yaitu:

Gambar 12. Data Pengujian *Performance Website* Frizfoo Menggunakan Lighthouse Sebelum Implementasi PWA

Gambar 13. Data Pengujian *Performance Website* Frizfoo Menggunakan Lighthouse Setelah Implementasi PWA

Berikut rincian aspek pengujian *performance website* Frizfoo menggunakan Lighthouse, seperti yang dapat dilihat pada Tabel 3 di bawah ini.

Tabel 3. Hasil Pengujian *Performance Website* Frizfoo Menggunakan Lighthouse
[Sumber : Hasil Pengujian *Performance Website* Frizfoo Menggunakan Lighthouse, 2023]

No.	Variabel	Sebelum Implementasi PWA	Setelah Implementasi PWA
1.	*Performance*	62%	98%
2.	*First Contentful Paint*	1.5s	0.9s
3.	*Total Blocking Time*	220ms	0ms
4.	*Speed Index*	2.3s	1.0s
5.	*Largest Contentful Paint*	2.8s	0.9s
6.	*Cumulative Layout Shift*	0.153	0.01

Berdasarkan Tabel 3, hasil pengujian *performance* pada *website* Frizfoo menggunakan Lighthouse setelah melakukan implementasi PWA dan beberapa optimasi (mengkonversi gambar ke format .webp, menyimpan *file* statis ke dalam *cache storage*, *update CDN* Font Awesome) menunjukkan peningkatan yang signifikan. Skor *performance* meningkat dari 62% menjadi 98%, dan berbagai variabel pengujian seperti yang terlihat pada Tabel 3 mengalami peningkatan yang signifikan. Hasil tersebut menunjukkan bahwa pengimplementasian teknologi PWA pada *website* Frizfoo, dapat meningkatkan *performance website*.

b. GTMetrix

Hasil pengujian *performance website* Frizfoo, baik dalam keadaan sebelum dan sesudah implementasi PWA menggunakan GTMetrix, yaitu:

Gambar 14. Data Pengujian *Performance Website* Frizfoo Menggunakan GTMetrix Sebelum Implementasi PWA

Gambar 15. Data Pengujian *Performance Website* Frizfoo Menggunakan GTMetrix Setelah Implementasi PWA

Berikut rincian aspek pengujian *performance website* Frizfoo menggunakan GTMetrix, seperti yang dapat dilihat pada Tabel 4 di bawah ini.

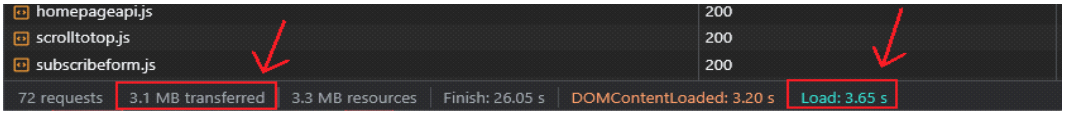
Tabel 4: Hasil Pengujian *Performance Website* Frizfoo Menggunakan GTMetrix
[Sumber : Hasil Pengujian *Performance Website* Frizfoo Menggunakan GTMetrix, 2023]

No.	Variabel	Sebelum Implementasi PWA	Setelah Implementasi PWA
1.	*Performance*	63% (*Grade* C)	99% (*Grade* A)
2.	*First Contentful Paint*	1.5s	174ms
3.	*Time to Interactive*	3.3s	488ms
4.	*Speed Index*	2.3s	432ms
5.	*Total Blocking Time*	77ms	63ms
6.	*Largest Contentful Paint*	2.7s	572ms
7.	*Cumulative Layout Shift*	0.31	0.08
8.	*Redirect Duration*	0ms	0ms
9.	*Time to First Byte (TTFB)*	667ms	40ms
10.	*Fully Loaded Time*	7.5s	1.6s
11.	*Total Page Size*	2.34 MB	2.30 MB
12.	*Total Page Requests*	52	53

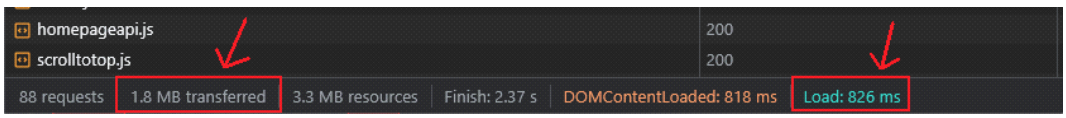
Berdasarkan Tabel 4, hasil pengujian *performance* pada *website* Frizfoo menggunakan GTMetrix setelah melakukan implementasi PWA dan beberapa optimasi (menghapus *tag script CDN* yang tidak terpakai, menambahkan atribut async pada *tag script*, minifikasi pada beberapa *file* css/javascript, dan sebagainya) menunjukkan peningkatan yang signifikan. Skor *performance* meningkat dari 63% menjadi 99%, dan terjadi peningkatan yang signifikan dalam hal kecepatan *loading* halaman *website*. Sebelum implementasi PWA, *website* membutuhkan waktu 7,5s untuk memuat satu halaman penuh (*Fully Loaded Time*), tetapi setelah melakukan implementasi PWA, waktu yang dibutuhkan hanya 1,6s.

4. Pengujian *Size Transferred Resources*

Hasil pengujian *Size Transferred Resources website* Frizfoo menggunakan *browser DevTools* dalam keadaan *disable cache*, yaitu:



Gambar 16. Data *Size Transferred Resources Website* Frizfoo Menggunakan *Browser DevTools* Sebelum Implementasi PWA

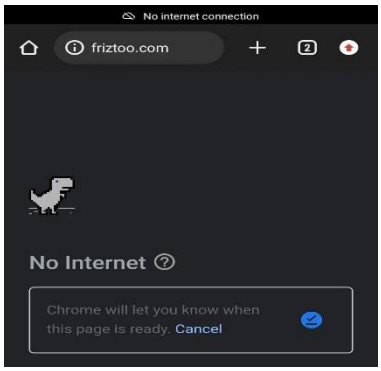


Gambar 17. Data *Size Transferred Resources Website* Frizfoo Menggunakan *Browser DevTools* Setelah Implementasi PWA

Hasil pengukuran *size transferred resources* pada *website* Frizfoo setelah melakukan implementasi PWA menunjukkan nilai sebesar 1.8 MB, dari yang sebelumnya sebesar 3,1 MB. Dengan demikian, pengguna perlu mengunduh sekitar 1.8 MB setiap ingin memuat halaman web tersebut. Selain itu, waktu *loading website* Frizfoo juga menjadi lebih cepat yaitu menurun dari 3,65 detik menjadi 826 ms setelah implementasi PWA. Sehingga implementasi PWA dapat mengurangi nilai *size transferred resources* serta mempercepat waktu *load website* Frizfoo.

5. Pengujian *Offline Mode*

Pengujian *Offline Mode* dilakukan untuk memastikan bahwa *website* Frizfoo telah berhasil mengadopsi salah satu fitur PWA yaitu *Offline Mode*. Pengujian dilakukan dengan menggunakan *browser DevTools* bagian *network* dengan mengganti bagian *no throttling* menjadi *offline* atau dalam kondisi tidak terhubung ke internet.



Gambar 18. Data Pengujian *Offline Mode Website* Frizfoo Sebelum Implementasi PWA

Gambar 19. Data Pengujian *Offline Mode Website* Frizfoo Setelah Implementasi PWA

Hasil pengujian *offline mode* pada *website* Frizfoo telah berhasil, karena *website* dapat diakses saat koneksi internet tidak tersedia atau tidak stabil, serta dapat menampilkan halaman statis khusus yang berisikan *user* sedang *offline*, sehingga diberikan nilai 1 sebagai indikasi keberhasilan *website* Frizfoo dalam pengujian *offline mode*.

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan oleh penulis, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Adanya peningkatan *performance website* Frizfoo setelah melakukan implementasi PWA. Hasil pengujian menggunakan Lighthouse mendapatkan hasil dari 62% sebelum implementasi PWA menjadi 98% setelah implementasi PWA. Sedangkan pengujian dengan menggunakan GTMetrix mendapatkan hasil dari 63% (*Grade* C) sebelum implementasi PWA menjadi 99% (*Grade* A) setelah implementasi PWA.
2. Penggunaan *framework* Express.js dalam implementasi PWA pada *website* Frizfoo memberikan kemudahan dalam hal fleksibilitas. *Framework* ini memungkinkan pengelolaan *template website* Frizfoo dengan mudah, mendukung *server side rendering* untuk

meningkatkan kinerja *performance website*, dan memberikan fleksibilitas dalam kostumisasi struktur folder serta fungsi *middleware*. Selain itu, Express.js juga menyediakan *built-in middleware* yang secara efisien menentukan folder statis mana yang dapat diakses agar dapat digunakan *file*-nya, khususnya *file* yang dibutuhkan dalam implementasi PWA.

3. Dengan implementasi PWA pada *website* Frizfoo, membuat pengguna dapat menginstal dan mengakses *website* seperti aplikasi *mobile native* atau aplikasi *desktop*, tanpa perlu menggunakan *browser* atau melakukan instalasi melalui Play Store atau App Store. Teknologi PWA berhasil diimplementasikan dan memenuhi seluruh kriteria yang diperlukan, dengan skor sempurna dalam pengujian menggunakan Lighthouse dan skor total 25 dengan 25 kriteria pengujian yang terpenuhi dalam pengujian menggunakan PWABuilder. Selain itu, terjadi penurunan *size transferred resources* serta waktu *load website* Frizfoo setelah melakukan implementasi PWA, dari yang sebelumnya 3.1 MB menjadi 1.8 MB dan waktu *load website* yang semakin cepat yaitu dari 3,65 detik menjadi 826 ms. Serta, dengan mengadopsi teknologi PWA, maka *website* Frizfoo dapat menggunakan fitur *offline mode* yang memungkinkan *website* dapat diakses dalam keadaan *offline* atau koneksi internet yang tidak stabil, dan memiliki kemampuan untuk menampilkan halaman statis khusus yang berisikan *user* sedang *offline*, saat pengguna mengakses *website* dalam keadaan *offline*.

DAFTAR PUSTAKA

[1] R. 'Eric, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, First Edition. New York: Crown, 2011. Accessed: May 29, 2023. [Online]. Available: https://www.google.co.id/books/edition/_/r9x-OXdzpPcC?hl=en&kptab=overview

[2] F. Wahyurianto, I. Arwani, and A. A. Soebroto, "Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang Berbasis Mobile Native Android," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 1, pp. 416–425, 2019, [Online]. Available: http://j-ptiik.ub.ac.id

[3] H. B. Putra and S. C. Wibawa, "STUDI LITERATUR PENGEMBANGAN E-COMMERCE SEKOLAH MENGGUNAKAN PROGRESSIVE WEB APPS (PWA)," 2021. Accessed: Nov. 17, 2022. [Online]. Available: https://ejournal.unesa.ac.id/index.php/it-edu/article/view/37931

[4] E. Budiman and M. Wati, "Pengembangan Progressive Web Application Portal Program Studi Teknik Informatika Berbasis Restful API," *Prosiding Seminar Nasional Ilmu Komputer dan Teknologi Informasi*, vol. 4, no. 2, 2019, Accessed: Dec. 14, 2022. [Online]. Available: https://core.ac.uk/reader/276528190

[5] A. Aminudin, B. Basren, and I. Nuryasin, "Perancangan Sistem Repositori Tugas Akhir Menggunakan Progressive Web App (PWA)," *Techno.Com*, vol. 18, no. 2, pp. 154–165, May 2019, doi: 10.33633/tc.v18i2.2309.

[6] A. Yuliana, R. Rigustama, and A. Zahra, "SISTEM INFORMASI PENILAIAN SEMINAR KERJA PROYEK DAN SIDANG TUGAS AKHIR DI POLITEKNIK TEDC BANDUNG," 2021. Accessed: Dec. 14, 2022. [Online]. Available: http://www.ejournal.poltektedc.ac.id/index.php/tedc/article/view/551

[7] S. Aripin and S. Somantri, “Implementasi Progressive Web Apps (PWA) pada Repository E-Portofolio Mahasiswa,” *Jurnal Eksplora Informatika*, vol. 10, no. 2, pp. 148–158, Mar. 2021, doi: 10.30864/eksplora.v10i2.486.

[8] J. Riady, H. N. Palit, and J. Andjarwirawan, “Aplikasi E-Learning Berbasis Progressive Web App Pada Apologetika Indonesia,” 2019. Accessed: Dec. 01, 2022. [Online]. Available: https://publication.petra.ac.id/index.php/teknik-informatika/article/view/8749

[9] C. C. B. Bahari and Y. Sumaryana, “Penerapan Progressive Web Apps Pada Aplikasi Lowongan Pekerjaan Dosen Universitas Perjuangan,” *Informatics and Digital Expert (INDEX)*, vol. 1, no. 1, pp. 25–31, Nov. 2019, doi: 10.36423/ide.v1i1.285.

[10] D. Haryanto and Z. Elsi, “Analisis Performance Progressive Web Apps Pada Aplikasi Shopee”, doi: http://dx.doi.org/10.36982/jiig.v12i2.1944.

[11] I. Aslan, H. Bahtiar, and A. Sudianto, “Pengembangan Website Fakultas Teknik Universitas Hamzanwadi Berbasis Progressive WEB APP (PWA),” *Infotek : Jurnal Informatika dan Teknologi*, vol. 5, no. 1, pp. 99–107, Jan. 2022, doi: 10.29408/jit.v5i1.4448.

[12] R. J. Phie Joarno, Mohammad Fajar, and Arfan Yunus, “Implementasi Progressive Web Apps Pada Website GetHelp Menggunakan Next.js,” *KHARISMA Tech*, vol. 17, no. 2, pp. 1–15, Sep. 2022, doi: 10.55645/kharismatech.v17i2.219.

[13] Chrome DevRel, “Web app manifest,” *web.dev*. https://web.dev/learn/pwa/web-app-manifest/ (accessed Jun. 05, 2023).

[14] Chrome DevRel, “Service workers.” https://web.dev/learn/pwa/service-workers/ (accessed Aug. 24, 2023).