BAB II
STUDI LITERATUR

2.1 Kajian Pustaka

Penelitian terdahulu yang terkait dengan pengembangan *Progressive Web Apps (PWA)* akan digunakan sebagai acuan dalam penelitian ini. Penelitian yang dilakukan oleh Wibowo [3] merancang kuisioner evaluasi mutu berbasis *mobile web* menggunakan PWA, menghasilkan sebuah *mobile web* yang dapat diakses dengan cepat karena menggunakan teknologi REST API untuk pertukaran datanya. *Mobile web* yang dihasilkan juga bersifat responsif dan dapat diakses secara *offline*.Pada penelitian ini juga dilakukan dua macam pengujian sistem yaitu pengujian REST API menggunakan Postman dan juga *response time* per halaman sistem menggunakan Chrome DevTools. Selanjutnya penelitian yang dilakukan oleh Adi [4] yang mengembangkan *platform e-Learning* untuk pembelajaran pemrograman web menggunakan konsep PWA dengan arsitektur *app shell*. Penelitian ini menghasilkan sebuah *platform e-Learning* yang dapat berjalan secara *offline* dimana pengguna tetap bisa mengakses soal – soal yang tersedia dan menyimpannya. Namun, masih belum dapat menyimpan jawabannya. Pada penelitian ini sistem diuji menggunakan Lighthouse untuk menguji *web app* dengan karakteristik dari PWA. Skor yang didapatkan dari pengujian dengan Lighthouse adalah 95,5 dari 100. Selain itu, sistem juga diuji menggunakan Chrome DevTools untuk melihat kinerja dari *service worker* yang diimplementasikan.

Penelitian yang dilakukan Nugroho, dkk. [5] yang mengembangkan sebuah sistem pengawasan untuk *smart farming* dengan menggunakan PWA untuk mengintegrasikan REST API yang didapat dari perangkat Internet of Things (IoT) dengan tampilan antarmuka pengguna. Hasil dari penelitian ini adalah sebuah sistem pengawasan yang dapat diakses oleh pekerjanya melalui *smartphone* atau *destkop*. Pada PWA ini juga terdapat fitur *push notification*. Sistem yang telah dibuat diuji menggunakan metode *black box*, *Lighthouse*, dan *PageSpeed Insight*. Skor yang didapatkan dari pengujian dengan *Lighthouse* 93,4 dari 100. Sedangkan untuk PageSpeed Insight adalah 87 dari 100.

Selanjutnya penelitian yang dilakukan oleh Loreto, dkk. [6] yang mengembangkan sebuah *Personal Health Record* (PHR) untuk mendukung wanita yang sedang hamil. Pengembangan dengan menggunakan PWA dipilih karena penulis ingin aplikasi yang dikembangkan dapat diakses menggunakan berbagai macam *device*. Pengembangan PWA menggunakan *framework* React untuk tampilan antarmukanya. Untuk pertukaran data menggunakan Express.js yang terhubung dengan basis data MongoDB. Penelitian ini menghasilkan sebuah *web app* yang dapat diakses secara *offline* dan bersifat responsif dimana *web app* akan menyesuaikan ukuran *device* yang mengaksesnya.

Penelitian yang dilakukan oleh Behl, dkk. [7] yang membahas mengenai arsitektur *app shell* dan *background synchronization* dalam PWA. PWA yang dibangun dengan arsitektur *app shell* memiliki tingkat waktu muat ulang yang selalu berkurang ketika PWA dikunjungi berkali – kali. Selanjutnya dengan *background synchronization* memungkinkan PWA untuk dapat menyimpan *request* ketika keadaan sedang *offline* dan meneruskan *request* dari *background* ketika sudah *online*. Selanjutnya penelitian dari Ridho, dkk. [8] yang melakukan perbandingan antara PWA dan *Mobile Web* terkait waktu respon, penggunaan memori dan media penyimpanan. Untuk melakukan perbandingan digunakan Google Chrome untuk melihat data hasil pengujian dan ADB Driver untuk menghubungkan Google Chrome dengan Chrome DevTools. Sedangkan pada perangkat bergerak digunakan Google Chrome untuk Android. Hasil dari perbandingan yang didapat adalah performa terkait waktu respon *mobile web* lebih unggul. Namun untk ukuran berkas dan *cache* cukup besar PWA lebih unggul. PWA juga lebih unggul jika diakses lebih dari satu kali. Performa terkait penggunaan memori *Mobile web* lebih kecil. Hal ini dikarenakan PWA memiliki *service worker*. Namun selisihnya tidak terlalu jauh yakni sekitar 2300 kB. Untuk performa terkait penggunaan media penyimpanan, *mobile web* lebih unggul karena tidak menggunakan ruang penyimpanan sama sekali.

Penelitian dari Gambhir, dkk. [9] yang melakukan analisis *cache* pada *service worker* terhadap performa dari PWA dan membandingkan performa dari aplikasi *Famous Birthdasy App* untuk PWA dan Android. Untuk menganalisis *service worker* digunakan Lighthouse dan untuk membandingkan performa

aplikasi digunakan Blazemeter dan dengan dua konfigurasi. Konfigutasi pertama *user* yang menggunakan aplikasi sebanyak satu dan konfigurasi kedua *user* yang menggunakan aplikasi sebanyak 50. Hasil yang didapatkan adalah *cache* pada *service worker* terbukti meningkatkan performa dikarenakan dengan melakukan *caching* meningkatkan rata – rata waktu respon. Untuk hasil perbandingan Android dan PWA berdasarkan hasil yang didapatkan semakin banyak jumlah *user* waktu rata – rata respon berkurang secara drastis. Sehingga PWA dapat dimuat dengan sangat cepat. Selanjutnya penelitian dari Aminudin, dkk. [10] yang merancang sistem repositori tugas akhir menggunakan PWA dengan arsitektur *app shell* sehingga membuat perpindahan halaman tidak memuat ulang halaman secara keseluruhan. Seluruh data statis yang diperlukan sistem disimpan pada *cache*s sehingga dapat diakses secara *offline* dan dapat dengan mudah dipasang pada *mobile* karena memunculkan *pop-up add to homescreen*. Untuk pertukaran data peneliti menggunakan Express.js dan basis data MongoDB. Sistem yang telah dibuat diuji mengunakan Lighthouse untuk menguji *web app* dengan karakteristik dari PWA. Skor yang didapatkan dari pengujian dengan Lighthouse adalah 92,4 dari 100.

Rangkuman penelitian terdahulu yang terkait dengan pengembangan PWA dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian terkait

No	Peneliti	Tahun	Judul	Hasil
1.	G. D. Wibowo	2017	Perancangan Kuisioner Evaluasi Mutu Berbasis Mobile Web Application Menggunakan PWA (Progressive Web App) (Studi Kasus : simutu.umm.ac.id)	Sistem evaluasi yang bersifat responsif dan dapat berjalan secara *offline* dan sebuah REST API untuk pertukaran data dalam sistem yang dibuat. Dilakukan dua pengujian yaitu menggunakan *Postman* untuk REST API dan juga

No	Peneliti	Tahun	Judul	Hasil
				Chrome DevTools untuk *response time.*
2.	L. Adi	2017	*Platform e-Learning* untuk Pembelajaran Pemrograman *Web* Menggunakan Konsep *Progressive Web Apps*	Sebuah *platform e-Learning* yang mampu menyimpan pertanyaan yang ada pada *course* sehingga dapat diakses secara *offline*. Dilakukan dua pengujian yaitu menggunakan *Lighthouse* dengan skor 95,5 dari 100 dan menggunakan *Chrome DevTools* untuk kinerja *service worker.*
3.	Nugroho L, Pratama A, Mustika I, Ferdiana R	2017	*Development of monitoring system for smart farming using Progressive Web App*	Sebuah sistem pengawasan untuk *smart farming* yang bersifat responsif dan dapat ditambahkan pada *homescreen* pengguna. Sistem diuji menggunakan *Lighthouse* dengan skor 93.4 dari 100 dan *Pagespeed Insight* dengan skor 87 dari 100.
4.	P. Loreto, J. Braga, H. Peixoto, J. Machado,	2018	*Step Towards Progressive Web Development in Obstetrics*	Sebuah sistem pencatat kesehatan pengguna untuk ibu hamil yang dapat berjalan secara

No	Peneliti	Tahun	Judul	Hasil
	and A. Abelha			*offline* dan bersifat responsif.
5.	Behl K, Raj G	2018	*Architectural Pattern of Progressive Web and Background Synchronization*	PWA dengan arsitektur *app shell* mengurangi waktu muat ulang ketika tiap kali dikunjungi. Dengan menggunakan *background synchronization* memungkinkan PWA untuk melakukan *request* pada *background process*
6.	Ridho M, Pinandito A, Dewi R	2018	Perbandingan Performa *Progressive Web Apps* dan *Mobile Web* Terkait Waktu Respon, Penggunaan Memori dan Penggunaan Media Penyimpanan	PWA memiliki performa terkait waktu respon yang lebih baik dibandingkan dengan *web app* jika ukuran berkas dan *cache* cukup besar dan diakses lebih dari satu kali. Penggunaan memori dan penyimpanan PWA lebih besar dikarenakan adanya *service worker* dan ukuran berkas *cache* yang disimpan.
7.	Gambhir A, Raj G	2018	*Analysis of Cache in Service Worker and Performance Scoring of*	Proses *caching* terbukti meningkatkan performa dari PWA dengan melakukan analisis dari hasil yang didapat dari

No	Peneliti	Tahun	Judul	Hasil
			Progressive Web Application	Lighthouse. Untuk aplikasi yang sama dari PWA dan Android terbukti performa PWA lebih baik.
8.	A. Aminudin, B. Basren, and I. Nuryasin	2019	Perancangan Sistem Repositori Tugas Akhir Menggunakan Progressive Web App (PWA)	Sebuah sistem repository yang bersifat responsif dan dapat berjalan secara *offline* dan terdapat *pop-up add to homescreen* untuk pemasangan aplikasi. Sistem diuji menggunakan *Lighthouse* dengan skor 92.5 dari 100.

2.2 Prakitkum

Menurut KBBI praktikum adalah bagian dari pengajaran yang bertujuan agar siswa mendapat kesempatan untuk menguji dan melaksanakan dalam keadaan nyata apa yang diperoleh dalam teori.

2.3 Pelaksanaan Praktikum Di Teknik Informatika

Dalam pelaksanaan praktikum pada program studi teknik informatika, dosen pengampu mata kuliah praktikum akan membagikan modul – modul pembelajaran sekaligus tugas kepada para asisten praktikum. Lalu asisten praktikum akan membagikan modul dan tugas tersebut kepada para peserta praktikum untuk tiap pertemuannya. Untuk pembagian modul kepada peserta praktikum dan pengumpulan tugas seluruhnya diatur oleh asisten praktikum mata kuliah tersebut.

2.4 Progressive Web Apps (PWA)

Menurut Mozilla Developer Network (2019) PWA adalah aplikasi *web* yang dikembangkan untuk memanfaatkan keunggulan dari aplikasi *web* dan aplikasi *native*. Sebagai contoh, aplikasi *web* lebih mudah untuk ditemukan karena akan lebih cepat untuk mengunjungi suatu *website* dibandingkan dengan memasang suatu aplikasi. Aplikasi *web* juga dapat dibagikan melalui *link*. Sedangkan keunggulan dari aplikasi *native* adalah lebih terintegrasi dengan sistem operasi sehingga pengalaman dari pengguna lebih mulus. Selain itu aplikasi *native* setelah dipasang dapat digunakan saat kondisi *offline* dan juga dapat dengan mudah diakses melalui ikon yang terdapat *homescreen* pengguna [11].

MDN juga menjelaskan aplikasi *web* harus memiliki beberapa karakteristik agar dapat disebut sebuah PWA. Adapun karakteristik dari PWA sebagai berikut :

a. *Discoverable*, aplikasi *web* dapat ditemukan melalui *search engines*.
b. *Installable,* aplikasi *web* dapat dipasang pada *homescreen* pengguna.
c. *Linkable,* aplikasi *web* dapat dibagikan dengan mengirim URL.
d. *Network independent,* aplikasi *web* dapat bekerja saat kondisi *offline* atau koneksi internet sedang buruk.
e. *Progressive,* aplikasi *web* tetap dapat berjalan pada *browser* lama dan berjalan secara maksimal di browser terbaru.
f. *Re-engageable*, aplikasi *web* dapat mengirimkan notifikasi ketika terdapat konten terbaru.
g. *Responsive,* aplikasi *web* dapat digunakan pada *device* dengan ukuran layar yang berbeda – beda seperti *mobile phone*, *tablets*, dan *destkop*.
h. *Safe,* koneksi antara pengguna dan aplikasi *web* terjaga dari pihak ketiga yang ingin mengakses data – data sensitif.

Terdapat dua arsitektur utama dari *rendering* sebuah *website* yaitu :

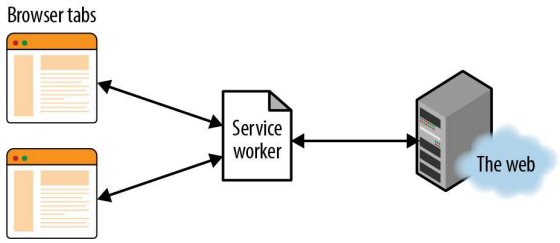
a. *Serverside–side rendering* (SSR), *website* akan di-*render* pada *server* sehingga saat mengunjungi pertama kali akan cepat. Namun saat berpindah halaman maka perlu mengunduh konten HTML yang baru.

b. *Client-side rendering* (CSR), *website* akan di-*update* di *browser* hampir secara instan ketika berpindah halaman. Saat mengunjungi *website* pertama kali lebih lambat, namun akan cepat saat berpindah halaman.

PWA dapat dikembangkan menggunakan arsitektur manapun, namun arsitektur yang populer digunakan adalah *Application Shell* dimana menggabungkan SSR dan CSR [12].

2.5 Service Worker

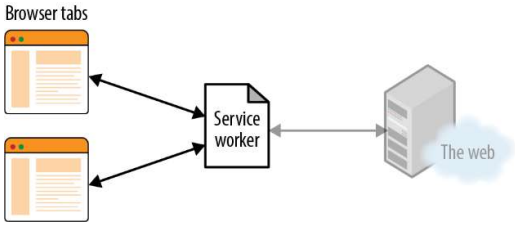
Ater (2017) dalam bukunya menjelaskan *service worker* adalah salat satu *web worker* yang berupa *file* JavaScript yang dapat di-*registered* untuk mengontrol halaman dari *website*. Setelah dipasang *service worker* berada di luar dari *browser*. Dari tempat ini service worker dapat mendengarkan dan melakukan aksi berdasarkan *events* yang terjadi dari halaman yang dikontrolnya seperti *request* untuk *file* dari *web* dapat diintersepsi, dimodifikasi, dioper dan dikembalikan ke halaman tersebut [13]. Seperti pada Gambar 2.1.



Gambar 2.1 *Service worker* saat kondisi *online*

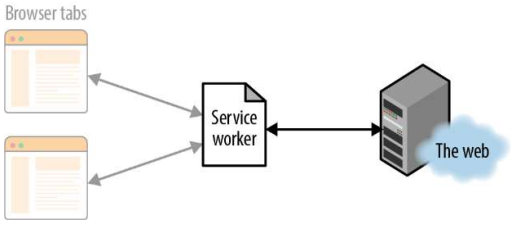
(Sumber : Buku *Building Progressive Web Apps: Bringing the Power of Native to the Browser,* 2017)

Hal ini membuat terdapat *layer* diantara halaman dan *web* yang dapat merespon secara independen dari koneksi internet. *Layer* yang dapat bekerja walaupun dalam keadaan *offline*. *Layer* ini dapat mendeteksi keadaan *offline* ataupun respon yang lambat dari *server* dan akan mengirimkan konten yang telah di-*cached* . Seperti pada Gambar 2.2.



Gambar 2.2 *Service worker* saat kondisi *offline*
(Sumber : Buku *Building Progressive Web Apps: Bringing the Power of Native to the Browser,* 2017)

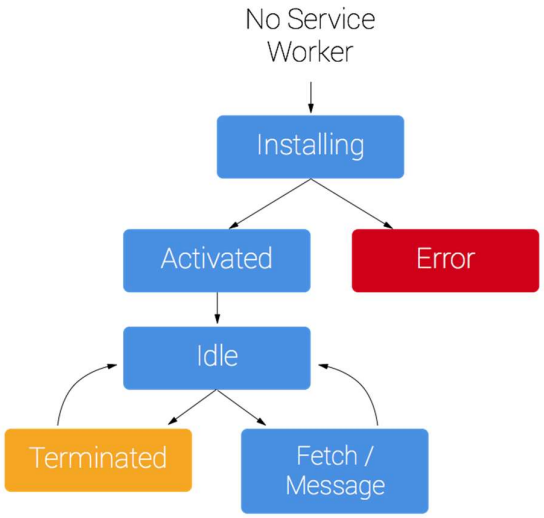
Walaupun pengguna menutup semua halaman yang ada di *browser*, *service worker* tetap memungkinkan untuk berkomunikasi dengan *server* seperti pada Gambar 2.3. *Service worker* dapat menerima dan menampilkan notifikasi atau memastikan aksi yang dilakukan oleh pengguna tersampaikan ke *server* .



Gambar 2.3 *Service worker* berjalan saat user menutup halaman
(Sumber : Buku *Building Progressive Web Apps: Bringing the Power of Native to the Browser,* 2017)

Service worker perlu didaftarkan (*registration)* untuk dapat berjalan pada sebuah *website*. Selanjutnya *browser* akan memulai tahap pemasangan (*instalation)* di latar. Pada tahap *instalation*, akan dilakukan *caching* dari aset – aset statis. Jika seluruh *file* berhasil di-*cached* maka *service worker* berhasil dipasang dan masuk ke tahap aktivasi (*activation)*. Jika terdapat *file* yang gagal untuk di-*cached* maka *service worker* tidak terpasang .

Setelah aktif, *service worker* akan mengontrol seluruh halaman pada *website* yang termasuk pada *scope* dimana *service worker* terpasang. Lalu, *service worker* akan berada pada salah satu kemungkinan *state*, yaitu *terminated* untuk menghemat memori, atau menangani *fetch* dan *message event* ketika terdapat *request* atau *message* dari halaman *web* . Diagram alur dari daur hidup (*lifecycle*) *service worker* dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Service worker lifecycle*
(Sumber : https://developers.google.com/web/fundamentals/primers/service-workers/images/sw-lifecycle.png)

Menurut Matt (2019) untuk mendapatkan akses *offline* pada sebuah *website*, *service worker* melakukan *caching* pada konten – konten *website*. Terdapat lima strategi *caching* yang paling umum digunakan untuk merespon suatu *request* yaitu [14] :

a. *Cache first, Network fallback* : *service worker* akan memuat HTML dan JavaScript dari *cache*. Jika kedua file tersebut tidak tersedia di *cache* maka *service* worker akan mengembalikan respon dari *network* dan menyimpannya kedalam *cache.* Strategi ini digunakan ketika menghadapi *resuorces* yang jarang berubah seperti Gambar statis.
b. *Network first, Cache fallback* : *service worker* akan melakukan *request* ke *network* terlebih dahulu dan jika berhasil mendapatkan respon maka *service worker* akan mengembalikan respon tersebut ke halaman. Jika *request* ke *network* gagal maka *service worker* akan mengembalikan data yang ada di dalam *cache*. Strategi ini digunakan ketika data yang diperlukan selalu baru namun tetap ingin menampilkan sebuah data ketika *network* sedang tidak tersedia.
c. *Cache/network race* : *service worker* akan melakukan *request* ke *network* dan *cache* secara bersamaan. Pada kebanyakan kasus, data yang berasal dari *cache* akan dimuat terlebih dahulu pada halaman. Sedangkan data yang

didapatkan dari *network* akan digunakan untuk membarui data yang ada didalam *cache*. Strategi ini digunakan ketika konten yang ingin ditampilkan diperbarui secara berkala seperti artikel, *timeline* sosial media dan peringkat pada *game*.

d. *Network only* : *service worker* hanya mengecek data pada *network*. Jika *request* pada *network* gagal, maka *request* gagal. Strategi ini digunakan ketika hanya data baru yang dapat ditampilkan pada halaman.

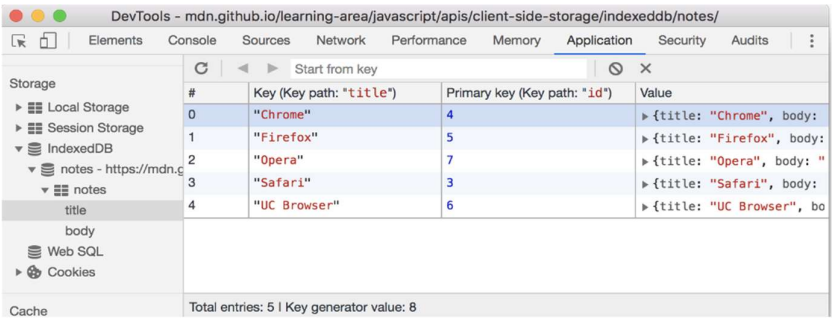
Cache only : *service worker* hanya menggunakan data yang di-*cached* pada saat pemasangan (*install) service worker*. Strategi ini digunakan ketika menampilkan data statis pada halaman.

2.6 Application Shell Architecture

Google Developer (2019) menjelaskan arsitektur *application shell* merupakan salah satu cara untuk membuat PWA dapat dimuat secara instan pada layar pengguna sama seperti pada aplikasi *native*. *Application shell* pada arsitektur ini merujuk pada *resources* lokal yang diperlukan oleh *web app* untuk memuat kerangka dari *user interface* (UI) seperti *headers, toolbars, footers* dan lain sebagainya. Hal ini membuat *application shell* berisi bagian dari halaman yang jarang berubah dan dapat di-*cached* sehingga dapat dimuat secara *instan* dari *cache* [15].

2.7 IndexedDB

Mozilla Developer Network (2019) menjelaskan IndexedDB merupakan *low-level* API untuk penyimpanan data dengan format terstruktur pada sisi *client*. API ini memungkinkan untuk membuat suatu *web app* dengan kemampuan *query* yang baik walaupun dalam kondisi *offline*. IndexedDB ini merupakan *transactional database system* mirip dengan RDBMS. Perbedaannya adalah IndexedDB berbasis *JavaScript Object* sedangkan RDBMS berbasis *table*. Sistem *indexing* digunakan dalam IndexedDB ini. IndexedDB dapat digunakan untuk menyimpan dan mengambil data berbentuk *object* dengan menggunakan suatu *key* [16]. Contoh dari penyimpanan IndexedDB dapat dilihat pada Gambar 2.5.



Gambar 2.5 Penyimpanan pada IndexedDB

2.8 App Manifest

App manifest adalah file JSON yang mengatur bagaimana sebuah *web app* akan ditampilkan. *App manifest* memungkinkan *browser* untuk menampilkan *banner* "*Add To Homescreen"*. Contohnya dapat dilihat pada Gambar 2.6.

```
{
  "short_name": "Maps",
  "name": "Google Maps",
  "icons": [
    {
      "src": "/images/icons-192.png",
      "type": "image/png",
      "sizes": "192x192"
    }
  ],
  "start_url": "/maps/?source=pwa",
  "background_color": "#3367D6",
  "display": "standalone",
  "scope": "/maps/",
}
```

Gambar 2.6 Potongan kode program app manifest

Adapun infromasi yang ada pada app manifest diantaranya :

1. *Short_name* / *name*

   Short_name merupakan nama yang akan digunakan pada *home screen* pengguna atau tempat dengan *space* sedikit. Jika *space* cukup akan menggunakan *name.*

2. *Icons*

   Icons merupakan kumpulan *icon* dengan berbagai ukuran yang akan digunakan pada aplikasi , *splash screen* dan lain sebagainya. Tiap *icon* memiliki *src* yang merupakan letak *icon* disimpan dan *sizes* yang merupakan ukuran dari *icon* tersebut.

3. *Start_url*

   Start_url merupakan URL yang perlu dijalankan ketika aplikasi dijalankan.

4. *Background_color*

   Background_color menentukan warna latar belakang dari *splash screen* ketika aplikasi dijalankan.

5. *Display*

   Display menentukan bagaimana aplikasi akan ditampilkan seperti menghilangkan *address bar*.

6. *Scope*

   Scope merupakan URL yang perlu ketika pengguna meninggalkan aplikasi.