

BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Awal Kurniawan (2017) Implementasi *Progressive Web Application* pada Sistem Monitoring Keluhan Sampah Kota Makassar. Pembuatan sistem ini menerapkan teknologi *Progressive Web Apps* (PWA) dan *service worker* yang membuat sistem akan membandingkan data pesanan dari client dengan data yang sudah ada di *server*. Ketika jaringan dalam keadaan aktif, maka aplikasi akan melakukan proses *caching file* dengan menyimpan data - data keluhan seperti nama pelapor, gambar keluhan, tanggal aduan, koordinat, dan keterangan keluhan.

Nurwanto (2019) Penerapan *Progressive Web Application* (PWA) pada *E-Commerce*. Penulis menggunakan *Content Management System*, dan ditambah dengan penggunaan teknologi *service worker* yang memungkinkan dapat dijalankan secara *offline* jika sebelumnya sudah dijalankan pada kondisi *online*.

I Bagus Prasetya Astina Putra (2020) Penerapan *Progressive Web Application* (PWA) Pada Aplikasi Manajemen Dokter Gigi. Fitur yang dihadirkan dapat mempermudah dokter dalam kegiatan operasional mulai dari pencatatan dan pencarian data pasien.

Riyanto Iqbal Firdaus (2016) merancang sistem media informasi berbasis *website* yang akan dikembangkan sehingga masyarakat dapat mengakses informasi pemerintah desa secara *online*. Informasi yang ditampilkan meliputi berita-berita, agenda kegiatan pemerintah desa, potensi desa, file - file yang dapat didownload, galeri foto, dan kontak dari pemerintahan desa yang dapat dihubungi. Bagi pemerintah desa, sistem ini akan meningkatkan pelayanan dari perangkat desa kepada penduduk desa terkait.

Hery Pujiwiyanto (2019) membuat aplikasi sistem informasi Desa Kerjo Kidul berbasis *website* dengan menggunakan *framework Codeigniter* dan *Bootstrap* yang menampilkan informasi profil desa, data penduduk, agenda kegiatan, dokumentasi dan kontak perangkat desa.

Tabel 2.1 Perbandingan Tinjauan Pustaka

Penulis	Objek	Teknologi	Hasil
Awal Kurniawan (2017)	Lab Pusat Teknologi Informasi dan Pangkalan Data Universitas Islam Negeri Alauddin	PWA	Proses menampilkan data keluhan di aplikasi dalam keadaan jaringan aktif, lalu proses menampilkan data keluhan di aplikasi dalam keadaan jaringan tidak aktif
Nurwanto (2019)	Web e-commerce	PWA	Dengan service worker memungkinkan e-commerce dapat diakses secara <i>offline</i> apabila sebelumnya pernah dijalankan secara <i>online</i> .
I Bagus Prasetya Astina Putra (2020)	Praktek DRG. Ketut Astina	PWA Mobile Web	Setiap halaman yang pernah dikunjungi oleh pengguna akan otomatis tersimpan pada <i>local storage browser</i> . Hanya saja user tidak dapat mengirim data input ke server.
Hery Pujiwiyanto (2019)	Desa Kerjo Kidul	Framework Codeigniter dan Bootstrap	Informasi Kependudukan dan agenda kegiatan desa.
Riyanto Iqbal Firdaus (2016)	Desa di Kecamatan Kedawung Sragen	Website	Menampilkan informasi-informasi dari pemerintah desa yang dapat dihubungkan dengan beberapa fitur tambahan.
Almas Suherwan	Desa di Kabupaten Kulon Progo	PWA	Informasi mengenai berita olahraga, data atlet, data pelatih, dan data cabang olahraga.

2.2 Dasar Teori

2.2.1 Progressive Web Apps

Teknologi aplikasi web saat ini sudah banyak mengalami perubahan fungsi. Awal mula perkembangan teknologi web dimulai dari web 1.0 yang diperkenalkan tahun 1990 dimana masih bersifat statis hingga menjadi aplikasi web yang dapat menangani masalah pengecekan status baterai, penggunaan mode *offline*, hingga *speech recognition*. Salah satu teknologi yang tengah banyak diperhatikan saat ini adalah teknologi *Progressive Web Apps* (PWA).

Progressive Web Apps (PWA) adalah istilah aplikasi berbasis web yang memanfaatkan fitur perambanan modern agar tampil seperti aplikasi *native*. PWA digambarkan sebagai penengah antara aplikasi *native* dan *hybrid* dengan berbagai rekomendasi yang tidak spesifik pada desain aplikasi web untuk perangkat *mobile*, seperti preferensi *https* melalui *http* dan desain yang *responsive*.

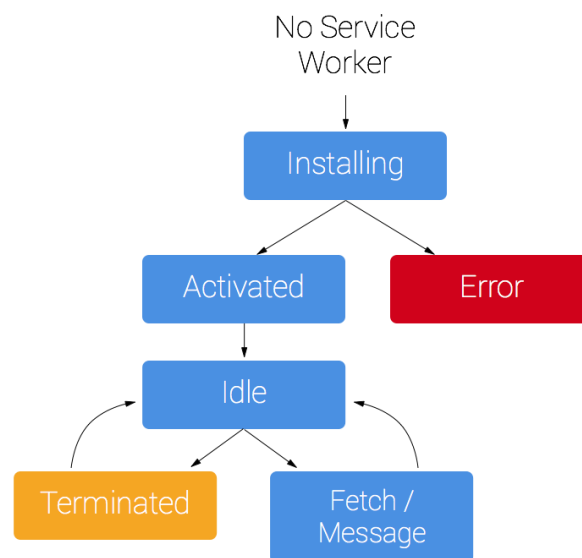
Pada dasarnya teknologi *Progressive Web Apps* (PWA) bekerja layaknya aplikasi berbasis web pada biasanya, yang membedakan PWA dengan aplikasi *website* lainnya adalah PWA yang bekerja dengan konektivitas yang independen. Artinya, aplikasi *Progressive Web Apps* dapat bekerja pada jaringan yang lemah atau dalam keadaan *offline* dengan adanya *service worker*. Dan selain mampu berjalan dalam keadaan *offline Progressive Web Apps* juga menggunakan teknologi *instant loading* yang membuat aplikasi *website* tersebut berjalan dengan cepat, *screenhome* dimana aplikasi *website* tersebut dapat dijadikan *icon* pada *desktop* atau *homescreen*.

Kata *prototype* berasal dari Bahasa Latin, yaitu “*proto*” yang berarti asli dan “*typus*” yang berarti bentuk atau model. Jadi *prototype* adalah versi simulasi produk akhir yang digunakan dalam pengujian terlebih dahulu sebelum akhirnya dilakukan peluncuran.

2.2.2 Service Worker

Salah satu konsep yang dibangun oleh *Progressive Web Apps* adalah *Service Worker*. *Service worker* adalah *script* yang berjalan di *background browser* pengguna. *Service worker* tidak membutuhkan sebuah halaman ataupun interaksi dari pengguna untuk menjalankan tugasnya, dengan begitu *service worker* akan terus berjalan walaupun halaman *web* tidak terbuka.

Service worker pada dasarnya adalah berkas *JavaScript* yang berjalan pada *thread* yang berbeda dengan *main thread browser*, menangani *network request*, *caching*, dan mengembalikan *resource* dari *cache*, dan bisa mengirim *push message*. *Service worker* bekerja sebagai pengatur *event fetch* dari *browser*, lalu *service worker* memutuskan apakah *request* atau permintaan akan diteruskan ke *server* atau ke *cache* berdasarkan kondisi jaringan *online* atau *offline* (Gaunt, 2019).



Gambar 2.1 Siklus Service Worker

Langkah - langkah dalam melakukan konfigurasi dasar *service worker* sebagai berikut :

1. Daftarkan *service worker* melalui URL (*Uniform Resource Locator*) fungsi `serviceWorkerContainer.register()`.
2. Jika berhasil, *service worker* dijalankan di `ServiceWorkerGlobalScope`.
3. *Service worker* telah siap untuk memproses *event*.

4. Instalasi *service worker* dicoba ketika *service worker* mengontrol halaman yang diakses setelah dan sebelumnya. *Event install* akan selalu dikirim pertama kali ke *service worker*.
5. Ketika *handler oninstall* selesai, *service worker* akan dipasang.
6. Proses aktivasi. Ketika *service worker* terpasang, selanjutnya akan menerima *event activate*. Pengguna utama dari *onactivate* ini adalah untuk membersihkan sumber daya yang digunakan sebelumnya.
7. *Service control* sekarang dapat mengatur halaman, tapi hanya dibuka setelah *register()* telah sukses seperti dokumen mulai aktif dengan atau tanpa *service worker* dan menjaganya selama masih digunakan.

2.2.3 IndexedDB

IndexedDB adalah sistem basis data transaksional, seperti RDBMS berbasis SQL. Namun, tidak seperti RDBMS berbasis SQL yang menggunakan tabel kolom tetap, *IndexedDB* adalah *database* berorientasi objek berbasis *JavaScript*. *IndexedDB* memungkinkan penyimpanan dan pengambilan objek yang diindeks dengan *key*, objek apa pun yang didukung oleh algoritma klon terstruktur dapat disimpan. Yang diperlukan ialah menentukan skema *database*, membuka koneksi ke *database*, dan kemudian mengambil dan memperbarui data dalam serangkaian transaksi. *IndexedDB* adalah cara untuk terus menyimpan data di dalam *browser* pengguna. Karena memungkinkan untuk membuat aplikasi *web* dengan kemampuan *query* yang kaya terlepas dari ketersediaan jaringan, aplikasi ini dapat berfungsi baik *online* maupun *offline*. *IndexedDB* sendiri memiliki kapasitas yang berbeda untuk setiap *browser*, diantaranya yaitu pada *Chrome* menyediakan sebanyak 6% dari *free space memory*, *Mozilla Firefox* menyediakan sebanyak 10% dari *free space memory* dan pada *Internet Explorer* sebanyak 250 megabyte. *IndexedDB* berguna

untuk aplikasi yang menyimpan sejumlah besar data (misalnya, katalog DVD di perpustakaan peminjaman) dan aplikasi yang tidak memerlukan konektivitas internet terus - menerus untuk bekerja (misalnya, *client email*, daftar tugas, dan buku catatan) (Walton, 2019).

2.2.4 Web App Manifest

Web Apps Manifest adalah file *JSON* sederhana yang memberitahu *browser* tentang aplikasi *web* dan bagaimana aplikasi itu dipasang pada perangkat *mobile* atau *desktop*. Diperlukan manifest oleh *Chrome* untuk menampilkan permintaan *Add To Home Screen* (A2HS). File manifest mencakup informasi tentang nama aplikasi, ikon yang harus digunakan, *start url* dimana manifest harus dimulai saat diluncurkan, dan banyak lagi (Gaunt & Kinlan 2019).

Saat pengguna menambahkan *website* ke layar beranda, sistem dapat menentukan serangkaian ikon untuk digunakan *browser*. Ikon ini digunakan di tempat - tempat seperti layar beranda, layar pembuka, dll. Ikon adalah *array* dari objek gambar. Setiap objek harus menyertakan *src*, *sizes property*, dan tipe gambar.