BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Beberapa penelitian yang menggunakan teknologi *Progressive Web Apps* untuk membangun *web*, penelitian ini menjadi acuan untuk membangun *web* penjualan rokok elektrik *(vape*).

Afif Rizki Kurniawan (2018) dalam penelitiannya yang berjudul "Penerapan *Progressive Web Apps* pada aplikasi lowongan pekerjaan dengan teknologi *service worker* (studi kasus akakom career center)" mencoba menerapkan teknologi *Progressive Web Apps* yang berfungsi untuk membuat *website* yang responsif dan dapat dijalankan dalam jaringan yang buruk. Penelitian ini menghasilkan sebuah sistem informasi lowongan pekerjaan yang *user-friendly* dan memungkinkan pelamar untuk melihat informasi lowongan pekerjaan secara *offline*.

Awal Kurniawan, Intan Sari Areni, Andani Achmad (2017) dalam penelitian mereka yang berjudul "Implementasi *Progressive Web Apps* pada sistem monitoring keluhan sampah Makassar" menekankan pada proses *caching file* pada konten *website* sehingga keluhan dapat ditampung bila jaringan sedang tidak aktif. Teknologi yang digunakan adalah *service worker* yang merupakan bagian dari *Progressive Web Apps*. Hasil penelitian ini memungkinkan data keluhan ditampilkan pada aplikasi dalam keadaan jaringan tidak aktif.

Laurensius Adi, Rizky Januar Akbar, Wijayanti Nurul Khotimah (2017) dalam penelitian berjudul "Platform E-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps" mencoba menerapkan teknologi

Progressive Web Apps untuk membuat *platform E-Learning* yang stabil dalam koneksi internet yang minim atau *offline*. Penelitian ini menghasilkan peningkatan performa *platform e-Learning*, terutama dalam waktu pemuatan halaman yang lebih cepat dan dapat berjalan secara *offline*.

Johan Putra Rahmadan (2020) dalam penelitiannya yang berjudul "Penerapan Progressive Web Apps untuk Web Organisasi Kampus di *Platform as a Service*" mencoba menerapkan teknologi *Progressive Web Apps* pada pengelolaan data alumni untuk program kerja organisasi. Penelitian ini menghasilkan sebuah sistem dengan teknologi *Progressive Web Apps* untuk web organisasi kampus menggunakan *Platform as a Service*.

Samsul Aripin, Somantri (2021) dalam penelitiannya yang berjudul "Implementasi Progressive Web Apps (PWA) pada Repository E-Portofolio Mahasiswa" mencoba menerapkan teknologi *Progressive Web Apps* pada sebuah sistem yang mengelola E-Portofolio mahasiswa. Penelitian ini menghasilkan sebuah repository E-Portofolio yang mendokumentasikan seluruh prestasi dan keaktifan mahasiswa selama perkuliahan di sebuah perguruan tinggi.

Tabel 2.1 Tinjauan Pustaka

Sumber	Judul	Teknologi	Masalah	Hasil
Afif Rizki Kurniawan (2018)	Penerapan *Progressive Web Apps* pada aplikasi lowongan pekerjaan dengan teknologi *service*	Progressive Web Apps,Service Worker	Penerapan Teknologi Service Working	Menghasilkan sebuah sistem informasi lowongan pekerjaan secara user friendly dan pelamar dapat melihat informasi lowongan pekerjaan secara offline

	worker (studi kasus akakom *career Center*)			dengan adanya service worker yang bekerja di backend
Awal Kurniawan, Intan Sari Areni,Andani Achmad (2017)	Implementasi Progressive Web Apps pada sistem Monitoring Keluhan Sampah Makassar	Progressive Web Apps	Jaringan yang tidak stabil untuk mengakses info keluhan sampah	Proses menampilkan data keluhan di aplikasi dalam keadaan jaringan internet aktif,lalu proses menampilkan data keluhan di aplikasi dalam keadaan jaringan tidak aktif
Laurensius Adi, Rizky Januar Akbar,Wijayanti Nurul Khotimah (2017)	Platform E-Learning untuk Pembelajaran Pemrograman web Menggunakan Konsep Progressive Web Apps	Progressive Web Apps	Ketersediaan Platform e-Learning yang tidak stabil dalam koneksi internet yang minim atau kondisi offline	Penerapan konsep PWA khususnya service worker meningkatkan performa platform eLearning terutama waktu memuat halaman menjadi lebih cepat dan dapat berjalan secara online
Johan Putra Rahmadan (2020)	Penerapan Progressive Web Apps untuk Web Organisasi Kampus Di Platform As A Service	Progressive Web Apps,PaaS	Pengelolaan data alumni untuk program kerja organisasi	Mengimplementasikan Progressive Web Apps untuk web organisasi Kampus menggunakan Platform As A Service
Samsul Aripin, Somantri (2021)	Implementasi Progressive Web Apps (PWA) pada Repository E-Portofolio	Progressive Web Apps	Membutuhkan sebuah sistem yang mengelola E-Portofolio mahasiswa	Menghasilkan sebuah repository E-Portofolio yang akan mendokumentasikan seluruh prestasi dan keaktifan mahasiswa

	Mahasiswa			selama perkuliahan di sebuah perguruan tinggi.
Farhan Rizky Aditya (2024)	Implementasi teknologi *progressive web apps* pada sistem penjualan rokok elektrik (vape) berbasis *paas cloud computing* (studi kasus : bariss vape corner**)**	Progressive Web Apps, PaaS	membangun sistem penjualan rokok elektrik	Mengimplementasikan Teknologi PWA dan PaaS Cloud Computing,dalam meningkatkan aksesibilitas dan kinerja sistem penjualan rokok elektrik vape, yang mendukung akses offline dengan penyimpanan cache.

2.2 Dasar Teori

2.2.1 Framework

Framework merupakan bentuk aturan tertentu yang berisi perintah atau fungsi dasar yang saling berinteraksi dalam pembuatan *website.* (Wardana, 2010)

Framework adalah kerangka kerja yang berisi perintah kode program dan fungsi untuk menjalankan tugas yang siap digunakan. (Purbadian, 2016)

Framework adalah *software* kerangka kerja untuk memudahkan dalam pembuatan aplikasi *web* agar tersusun dan terstruktur. (Roza, Fauzan, & Rahayu, 2020)

Dari definisi di atas dapat disimpulkan bahwa *framework* adalah kerangka kerja yang siap pakai, berisi kumpulan perintah atau fungsi yang saling berinteraksi untuk memudahkan pembuatan *website.*

2.2.2 Laravel

Laravel adalah sebuah *framework web* berbasis PHP yang *open-source* dan tidak berbayar, diciptakan oleh Taylor Otwell dan diperuntukkan untuk pengembangan aplikasi *web* yang menggunakan pola MVC. Struktur pola MVC pada laravel sedikit berbeda pada struktur pola MVC pada umumnya. Di *laravel* terdapat *routing* yang menjembatani antara *request* dari *user* dan *controller*. Jadi *controller* tidak langsung menerima *request* tersebut (Yudanto dkk, 2017).

Menurut (Wardana, 2014), *Model View Controller* (MVC) adalah sebuah pola pemrograman yang bertujuan memisahkan logika bisnis, logika data dan logika tampilan (interface), atau secara sederhana memisahkan antara proses, data, dan tampilan. MVC mengatur arsitektur sebuah aplikasi, umumnya aplikasi yang dibangun dengan konsep MVC adalah aplikasi yang cukup besar, karena salah satu keuntungan menggunakan konsep MVC adalah kemudahan untuk maintenance dan pengembangan aplikasi tersebut. *Laravel* 23 menggunakan konsep MVC, yang mana anda harus memisahkan kode database ke folder model, kode proses ke folder controller dan kode tampilan ke folder view. MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen-komponen aplikasi, seperti: manipulasi data, *controller*, dan *user interface*.

a. *View* adalah komponen dari MVC yang bertugas menampilkan apa yang harus ditampilkan ke pengunjung website. Isinya dapat berupa form, tabel, gambar, animasi ataupun lainnya yang dapat berupa *form*, tabel, gambar, animasi ataupun lainnya yang boleh dilihat oleh *user* (pengguna). Jadi, *view* mengatur bagaimana

suatu data yang diperoleh dari controller ditampilkan untuk user. *View* mencakup semua proses yang terkait *layout output*. Tempat menaruh *template interface website* atau aplikasi. *View* merupakan informasi yang ditampilkan kepada pengunjung dari *website*.

b. *Model* adalah komponen MVC yang bertugas mengambil data dari *database* dan juga memasukkan data ke database. Isi utamanya berupa perintah *SQL*. Hasilnya dikirimkan ke *Controller*.

c. *Controller* adalah komponen MVC yang bertugas mengirim perintah ke *model* untuk mendapatkan data yang diinginkan. *Controller* tidak mengetahui bagaimana data tersebut diambil dari database, karena *Controller* tidak berisi kode perintah *SQL*. Karena itu adalah tugas modal, *Controller* mengolah data dari masukkan user dan data dari modal kemudian data olahan tersebut dikirimkan ke view untuk ditampilkan sesuai aturan *controller*. *Controller* merupakan penghubung antara model dan *view* dan mengatur hubungan tersebut.

2.2.3 Lighthouse

Lighthouse adalah alat audit *open source* otomatis untuk meningkatkan kualitas halaman *web*. Alat ini memberikan cara yang jelas untuk meningkatkan kualitas situs dengan memungkinkan developer menjalankan audit untuk *Performance, Accessibility, Best Practice, SEO* dan *Progressive Web Apps*. Pada dasarnya, alat ini "menjaga agar tidak mengalami masalah", karena itu diberi nama *Lighthouse. Lighthouse* dikembangkan oleh Google dan dapat digunakan pada browser seperti *google Chrome, Firefox dan Brave*. Seperti yang disebutkan diatas

terdapat 5 metrik dari *Lightouse* yaitu *Performance*, *Accessibility*, *Best Practice*, *SEO* dan PWA.

Semua metrik memiliki penilaian dari 0 sampai 100. Nilai 0-49 berarti buruk, Nilai 50-89 berarti cukup dan nilai 90-100 adalah yang paling baik.

Terdapat deskripsi dari masing masing metrik diatas :

1. *Perfromance*

   Secara garis besar metrik ini mengukur seberapa cepat situs yang dapat ditampilkan pada pengguna saat mengaksesnya, skor metrik *Performance* ini sendiri terdiri dari enam metrik turunan yaitu :

   a. *First Contentful Paint*: Waktu yang dibutuhkan untuk gambar pertama di-*render*.
   b. *Time to Interactive*: Waktu hingga pengguna bisa berinteraksi dengan suatu halaman.
   c. *Total Blocking Time* : Waktu antara *First Contenful Paint* dan *Time to Interactive*. Pada moment ini user bisa melihat halaman, tapi belum bisa berinteraksi.
   d. *Largest Contenful Paint* : Waktu yang dibutuhkan untuk teks atau gambar terbesar di-*render.*
   e. *Speed Index*: Waktu yang dibutuhkan untuk teks atau gambar terbesar di-*render*.
   f. *Cumulative Layout Shift* : total pergerakan layout yang terjadi dihalaman situs.

2. *Accessibility*

   Accessibility adalah penilaian aksebilitas halaman bagi user dengan keterbatasan fisik maupun kognitif, untuk penilaian aksesbilitas ini, *Google Lightouse* akan mengecek *HTML*, *tags*, *ALT*,*text*, *ARIA landmark*.

3. *Best Practice*

   Google Lighthouse menganalisis penggunaan *best practice* atau cara-cara terbaik untuk membangun sebuah situs. Yang dinilai adalah penggunaan *HTTPS* untuk keamanan, *outbound link* yang aman, dan *javascript* yang terbaru.

4. *SEO*

   SEO adalah penilaian *Lighthouse* yang menilai seberapa patuh konten terhadap aturan-aturan algoritma *SEO* yang berlaku.

5. PWA

   PWA adalah standar yang ditetapkan oleh *Google*. Salah satu kriteria adalah apakah suatu situs dapat diakes dari lokasi dan device mana pun. di PWA ini tidak menampilkan skor, tetapi *google* akan memberi penilaian apakah web sudah memenuhi kriteria atau tidak.

2.2.4 Progressive Web Apps

Progressive Web Apps (PWA) adalah konsep pengalaman pengguna yang menghubungkan bagian terbaik web dan bagian terbaik *native apps*. PWA berguna bagi pengguna sejak pertama membuka halaman sebuah web dengan konsep PWA, dan seiring dengan pengguna menggunakan aplikasi web lebih banyak lagi, aplikasi

akan menjadi semakin powerful, Aplikasi dapat dimuat dengan cepat, bahkan dalam kondisi internet yang kurang baik, bias mengirim push notifications, punya ikon aplikasi di home screen, dan bias berjalan dalam mode layar penuh, Sebuah aplikasi PWA memiliki karakteristik sebagai berikut.

1. *Progressive*: berjalan dengan baik bagi semua pengguna, tidak memandang browser yang dipakai, karena dibangun dengan progressive enchancement dari sebuah inti aplikasi.
2. *Responsive*: ditampilkan dengan baik pada semua perangkat berbagai ukuran : desktop, tablet, mobile, dan apapun perangkat baru selanjutnya.
3. *Connectivity independent*: dengan *service worker* aplikasi dapat bekerja dalam kondisi *offline* atau jaringan lemah.
4. *App-like*: terasa seperti sebuah aplikai, karena *model app shell* yang memisahkan fungsionalitas aplikasi dari kontennya.
5. *Fresh* : selalu *up-to-date* nerkat pembaruan melalui *service worker.*
6. *Safe*: aplikasi dilayani via *HTTPS* untuk mencegah *snooping* dan memastikan konten tidak diubah.
7. *Discoverable*: dapat diidentifikasi sebagai”application” karena *manifest W3C* dan *service worker registration scope*, dan memunugkinkan mesin pencari untuk menemukannya.
8. *Re-engageable*: memudahkan dalam mengajak pengguna untuk menggunakan ulang aplikasi melalui fitur seperti *push notifications*.
9. *Installable* : memungkinkan pengguna untuk menambahkan aplikasi ke *homescreen* tanpa harus kerepotan dengan *app store*.

10. *Linkable* : dengan mudah dapat membagikan aplikasi dengan membagikan *URL* dan tidak memerlukan instalasi yang rumit.

Dalam membangun PWA ada beberapa komponen yang harus digunakan. Berikut adalah penjelasan ringkas dari beberapa komponen PWA yang akan kita pelajari. Ada beberapa komponen wajib dan ada juga komponen *optional* . Komponen wajib adalah komponen tang selalu digunakan setiap kali membuat PWA, sedangkan komponen opsional adalah komponen yang tidak mempengaruhi kinerja PWA namun dapat digunakan untuk memperkaya fitur PWA. Berikut komponen wajib yang dimaksud.

1. *HTTPS*: PWA harus dihosting di server yang menggunakan *HTTPS* untuk memastikan keamanan data yang dikirimkan antara server dan klien.
2. *Web App Manifest*: sebuah file *JSON* yang mendeskriptikan aplikasi web, seperti nama,ikon,warna tema, dan pengaturan layar awal. File ini memungkinkan pengguna utuk menginstal PWA ke layar utama pernagkat dengan pengalaman seperti aplikasi asli.
3. *Service Workers*: Skrip yang berjalan di latar belakang dan mengelola caching, memungkinkan aplikasi berfungsi *offline*, dan memberikan notifikasi *push*.

2.2.5 Platform As A Service (PaaS)

Platform as a service (PaaS) adalah salah satu jenis layanan dari cloud computing, PaaS merupakan layanan perangkat lunak perantara pada lingkungan cloud untuk memfasilitasi berjalannya program aplikasi-aplikasi lainnya. Layanan ini untuk membangun, menguji dan meyebarkan aplikasi yang sedang dalam tahap

pengembangan. PaaS merupakan layanan yang menyediakan hardware sehingga pengembangan aplikasi tidak perlu memikirkan *operating system, infrastructure scalling, load balancing* dan lainnya. Pengembang dapat focus pada aplikasi yang akan dikembangkan karena "tempat" untuk aplikasi sudah menjadi tanggung jawab *provider*. (Yunie, 2014)

Ika (2017) layanan *cloud* pada jenis ini disediakan dalam bentuk *platform* yang dapat dimanfaatkan pengguna untuk membuat aplikasi di atasnya . Hal-hal yang dapat dilakukan pengguna layanan PaaS adalah membangun aplikasi, upload aplikasi, testing, dan mengatur konfigurasi. Teknologi PaaS yang digunakan pada penelitian ini adalah Dewacloud. Teknologi ini mempermudah proses dalam *deployment* dan *maintenance* suatu *service.*

2.2.6 Service Worker

Service worker merupakan sebuah berkas *JavaScript* yang diproses oleh browser di latar belakang. Berkas ini dieksekusi secara terpisah tak seperti berkas javascript biasa yang membentuk website: *Service Worker* menjadi gerbang bagi berbagai fitur browser yang tidak memerlukantampilan atau interaksi dengan pengguna. Kemampuan utama dari service worker adalah mengambil alih seluruh urusan request pada browser.

Dengan adanya *Service Worker* sebagai perantara *(proxy)* kita dapat menerapkan sebuah logika sebelum request yang dilakukan browser benar-benar dikirimkan ke server, begitu pula sebaiknya. Contohnya, kita dapat menyimpan response yang didapatkan dari web server ke dalam cache API sebelum datanya ditampilkan pada jendela browser, contohnya lain kita bias mengevaluasi sebuah

request apakah sudah terapat pada *cache* atau belum, sebelum dikirimkan ke server. Dengan *service worker*, kita dapat melakukan apapun terhadap *request* dan *response* yang terjadi pada browser. Karena *Service Worker* memegang posisi penting dalam jalur transaksi data, *Service worker* hanya dapat dijalankan pada protocol HTTPS. Pemaksaan ini bertujuan untuk meningkatkan keamanan aplikasi.

Bila tidak seperti itu, sangat rentan terhadap serangan yang disebut "man-in-the-middle". Namun selama proses pengembangan, *Service worker* dapat berjalan pada *localhost*.