BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Berikut merupakan beberapa sajian tentang penelitian serupa yang terkait dengan penelitian ini, diantaranya.

Abdul Hakim (2018). Melakukan penelitian tentang *Progressive Web Apps* Pada Organisasi Ikatan Mahasiswa Tanjungbalai Jogjakarta. Dimana penelitian tersebut, penulis membuat suatu aplikasi berbasis web yang ditujukan untuk organisasi IMTA jogja yang dapat diakses setiap anggota, teknologi yang dibangun menggunakan pendekatan teknologi *progressive web apps* yang dapat berjalan ketika internet tidak stabil bahkan *offline.*

Afif Rizki Kurniawan (2018) pada penelitian ini, peneliti membangun suatu aplikasi berbasis *progressive web apps* menggunakan *javascript* yang dapat membantu para pencari pekerjaan untuk mendapatkan informasi yang mudah dan cepat, teknologi yang diterapkan yaitu *service worker* yang berfungsi untuk menyimpan *cache* pada website yang nantinya dapat berjalan oleh *browser* dalam kondisi *offline* sekalipun.

Ridho dkk (2017) dalam jurnalnya mengenai perbandingan performa *progressive web apps* dan *mobile web* terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan pada suatu halaman web. Penggunaan memori, *mobile web* menggunakan memori lebih sedikit dibandingkan dengan

progressive web apps dikarenakan adanya proses tambahan pada *progressive web apps* yaitu *service worker*. Sedangkan untuk performa terkait media penyimpanan pada *mobile web* tidak menggunakan media penyimpanan sama sekali, sedangkan pada *progressive web apps* penggunaan media penyimpanan menyesuaikan dengan *cache* yang disimpan pada peramban.

Adi dkk (2017) dalam jurnalnya yang meneliti mengenai *Platform E-Learning* untuk pembelajaran pemrograman web menggunakan konsep *progressive web apps* yang dapat membantu proses pembelajaran di Teknik Informatika Institut Teknologi Sepuluh Nopember, Surabaya. Pada penelitian ini *Platform E-Learning* mampu menampilkan halaman *E-Learning* yang dapat diakses oleh mahasiswa.

Aditya Nur Johansyah (2013) dalam tugas akhirnya yang meneliti tentang bagaimana membangun sebuah website *e-commerce* dengan aman dan dapat menambah kepercayaan konsumennya. Pada penelitian ini website tersebut diintegrasikan dengan sistem pembayaran PayPal atau biasa disebut dengan *PayPal Payment Gateway* guna memberikan keamanan terhadap transaksi antar negara dan memberikan dampak terhadap perluasan pasar yang bisa dijangkau.

Abdul Hamid (2019) dalam penelitian ini, penulis membangun sebuah aplikasi berbasis web mobile yang berfokus pada penerapan teknologi *Payment Request API* untuk mempercepat proses transaksi dalam proses pembayaran yaitu dengan malakukan pembayaran langsung di aplikasi melalui sebuah panggilan *API* dengan menggunakan library stripe dan menerapkan teknologi *Progressive Web Apps* yang bertujuan agar aplikasi mampu diakses dengan cepat tanpa harus

mengakses melalui *browser* dan membuat aplikasi mampu menampilkan katalog barang walaupun dengan koneksi internet yang minim bahkan *offline*.

Tabel 2.1. Acuan Tinjauan Pustaka

Penulis	Objek	Metode	Bahasa Pemrograman	Interface
Abdul Hakim (2018)	Organisasi IMTA Yogyakarta	Progressive Web Apps	Javascript	GUI
Afif Rizki Kurniawan (2018)	ACC (AKAKOM Career Center)	Progressive Web Apps	PHP	GUI
Ridho dkk (2017)	Manajemen cache pada aplikasi,	Progressive Web Apps	-	Teks
Adi dkk (2017)	Mahasiswa ITS Jurusan TI	Progressive Web Apps	PHP	GUI
Aditya Nur Johansyah, 2013	PT. Ekko Hejo	Prototyping	-	GUI
Abdul Hamid (2019)	Penjualan Laptop di Eksekutif Komputer	Payment Request API, Progressive Web Apps	PHP dan Javascript	GUI

2.2 Dasar Teori

2.2.1 Payment Request API

Payment API adalah standar web yang dikembangkan oleh W3C untuk menyederhanakan pembayaran *online* dan memungkinkan sekelompok orang yang lebih luas untuk berpartisipasi dengan mudah dalam proses pembayaran di web. Standarnya fleksibel, *Payment API* bekerja dengan berbagai jenis sistem

pembayaran dan tidak terikat pada browser, metode pembayaran, atau penyedia layanan pembayaran tertentu. Fleksibilitas ini memungkinkan pengembangan yang sederhana, konsistensi, penyebaran, dan kompabilitas masa depan dengan teknologi pembayaran yang muncul. Adapun manfaat menggunakan *Payment Request API :*

- **Konsumen** – Menyederhanakan alur *checkout*, dengan membuatnya beberapa klik saja tanpa harus banyak mengetik untuk mengisikan data kedalam *form*.
- **Pedagang** – Membuat lebih mudah untuk diterapkan dengan berbagai opsi pembayaran yang sudah difilter untuk pelanggan.
- **Penangan pembayaran –** Memungkinkan membawa semua jenis metode pambayaran ke web dengan integrasi yang *relative* mudah.
- **Penyedia layanan pembayaran –** Membawa metode pembayaran baru dan meningkatkan kemampuan bisnis untuk melayani lebih banyak pelanggan dengan pengalaman pengembang yang lebih baik dan solusi lebih aman (Kitamura Eiji, Gash Dave, 2018).

2.2.2 Stripe

Stripe adalah perusahaan teknologi yang membangun infrastruktur ekonomi dalam menerima pembayaran melalui internet. Perusahaan ini menciptakan sebuah perangkat lunak yang bernama stripe untuk menerima pembayaran dan mengelola bisnis dari berbagai ukuran mulai dari individu hingga perusahaan secara online . Stripe menyediakan *API* yang dapat digunakan pengembang web untuk mengintegrasikan pemrosesan pembayaran ke dalam situs web atau aplikasi seluler.

Untuk menggunakan *API key* dari stripe dalam sistem pembayaran, maka dibutuhkan sebuah *library* dari stripe yang dibungkus dalam kelas dengan nama stripe. Terdapat dua jenis *API key* yang digunakan, diantaranya *published key* yang digunakan di sisi klien untuk menangani informasi pembayaran yang akan dikirim dan *secret key* yang digunakan di sisi server dalam menangani informasi pembayaran yang akan diterima.

Ada dua proses untuk menerapkan sistem pembayaran menggunakan stripe dengan tindakan di sisi klien dan sisi server, diantaranya :

1. Mengumpulkan informasi pembayaran dari klien dan memberi sebuah token dalam setiap transaksi. Kemudian dikirimkan oleh browser ke server. Ketika pelanggan memasukkan informasi pembayaran, rincian kartu tersebut divalidasi. Dalam sebuah form informasi pembayaran, maka akan disisipkan data token untuk setiap transaksi dengan menggunakan *pulished key*, pembuatan data token menggunakan kode *javascript* berikut.

```
var stripe = Stripe('pk_test_KKjCo10blSfDT55pQm6AYgps');

var elements = stripe.elements();

var form = document.getElementById('payment-form');

form.addEventListener('submit', function(event) {
  event.preventDefault();

  stripe.createToken(card).then(function(result) {
    if (result.error) {
      // Inform the user if there was an error
      var errorElement = document.getElementById(
                          'card-errors');
      errorElement.textContent = result.error.message;
    } else {
      stripeTokenHandler(result.token);
    }
```

```
  });
});
function stripeTokenHandler(token) {
  var form = document.getElementById('payment-form');
  var hiddenInput = document.createElement('input');
  hiddenInput.setAttribute('type', 'hidden');
  hiddenInput.setAttribute('name', 'stripeToken');
  hiddenInput.setAttribute('value', token.id);
  form.appendChild(hiddenInput);

  form.submit();
}
```

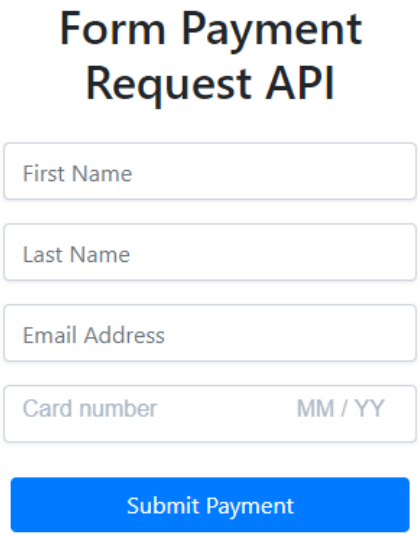
Dari *script* diatas, pembuatan data token dilakukan menggunakan fungsi *stripeTokenHandler*. Ketika form pembayaran diklik, maka fungsi tersebut akan membuat token dengan menambahkan form baru dengan *name stripeToken* dan *type hidden*. Sehingga form token tidak ditampilkan dibrowser. Kemudian untuk mengumpulkan informasi pembayaran, dapat dibuat menggunakan HTML. Seperti contoh berikut.

```
<h2 class="my-4 text-center">Form Payment Request API</h2>
<form action="./charge.php" method="post" id="payment-form">
  <div class="form-row">
    <input type="text" name="first_name"
       class="form-control mb-3 StripeElement
       StripeElement—empty placeholder="First Name">

    <input type="text" name="last_name"
       class="form-control mb-3 StripeElement
       StripeElement--empty" placeholder="Last Name">

    <input type="email" name="email"
       class="form-control mb-3 StripeElement
       StripeElement--empty" placeholder="Email Address">

     <div id="card-element" class="form-control"></div>
    <div id="card-errors" role="alert"></div>
   </div>
 <button>Submit Payment</button>
</form>
```



Gambar 2.1. Contoh Form Payment Request API

Dalam menangani informasi pembayaran yang akan dikirimkan ke sisi server, dibutuhkan beberapa data inti dari form diatas. diantaranya, nomor kartu (*card number*), *expired date* kartu, mata uang, dan total pembayaran. Kemudian bisa ditambahkan beberapa informasi tambahan seperti nama pembeli, email dan nama barang yang dibeli. Dari contoh form diatas ada beberapa data yang harus dimasukkan oleh pembeli yaitu nama pembeli, email, nomor kartu (*card number*) dan *expired date* kartu.

2. Dengan menggunakan *token* dan *published key* , maka di sisi *server* akan membuat permintaan *API key* berupa *secret key* untuk membuat biaya pembayaran. Dengan menggunakan PHP, maka untuk membuat pembayaran dengan stripe dapat dilakukan seperti potongan program berikut.

```
<?php

\Stripe\Stripe::setApiKey('sk_test_cN9vLaoub0RiPVp4CfdT');

 $POST = filter_var_array($_POST, FILTER_SANITIZE_STRING);

 $first_name = $POST['first_name'];
 $last_name = $POST['last_name'];
 $email = $POST['email'];
 $token = $POST['stripeToken'];

 $customer = \Stripe\Customer::create(array(
   "email" => $email,
   "source" => $token
 ));

 $charge = \Stripe\Charge::create(array(
   "amount" => 5000,
   "currency" => "usd",
   "description" => "Sepatu Baru",
   "customer" => $customer->id
 ));
?>
```

Setelah data diinputkan oleh pembeli, server akan membuat permintaan *API key* untuk membuat biaya pembayaran. Permintaan ini berisi token yang dibuat menggunakan fungsi *stripeTokenHandler*, mata uang (*currency*), jumlah (*amount*), dan informasi tambahan lainnya (*description*). Token hanya dapat digunakan satu kali dalam satu kali transaksi. Dengan menggunakan pendekatan ini, pelanggan perlu memasukkan kembali rincian pembayaran mereka setiap kali mereka melakukan pembelian untuk menjaga keamanan data (Developers Stripe, 2018).

2.2.3 Progressive Web Apps

Progressive Web Apps adalah sebuah pengalaman yang menggabungkan yang terbaik dari aplikasi web. Seiring dengan pengguna yang semakin berhubungan dengan aplikasi dari waktu ke waktu, *progressive web apps* menjadi

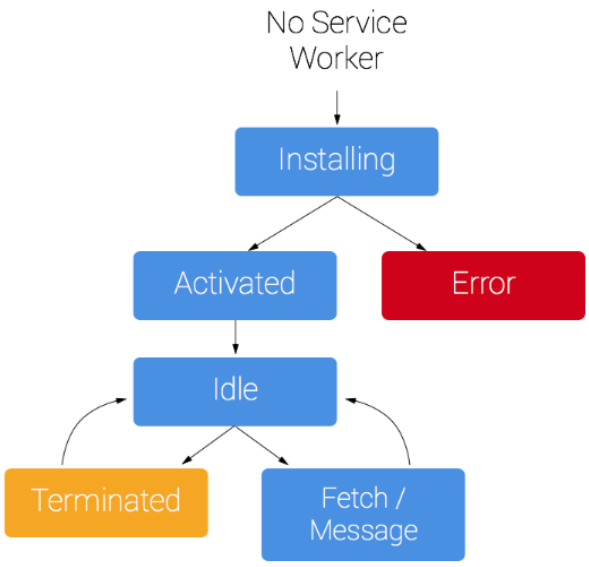
sangat *powerful.* Dapat memuat halaman dengan cepat, bahkan pada jaringan yang buruk sekalipun, mengirim *notifications* yang *relevan*, memiliki *icon* pada layar awal, dan memuat halaman sebagai pengalaman pengguna yang sangat baik. Adapun kelebihan dari teknologi *progressive web apps* antara lain :

- ***Progressive*** – Bekerja untuk setiap user, terlepas dari apapun jenis browsernya karena itu dibangun dengan *progressive* yang sangat baik
- ***Responsive*** – cocok dengan disegala *device* desktop, seluler, tablet, dan yang lainnya.
- ***Connectivity Independent*** – Ditingkatkan dengan *service worker* agar aplikasi bisa *offline* atau di jaringan berkualitas rendah sekalipun.
- ***App-like*** – Terasa seperti aplikasi, karena model shell aplikasi memisahkan fungsi aplikasi dari konten aplikasi.
- ***Fresh*** – Selalu diperbarui berkat pembaruan *service worker*.
- ***Safe*** – Dijalankan melalui *HTTPS* untuk mencegah pengintaian dan memastikan konten belum dirusak.
- ***Discoverable*** – Dapat diidentifikasi sebagai "aplikasi" berkat W3C *manifets* dan ruang lingkup *service worker*, yang memungkinkan mesin pencari menemukannya.
- ***Re-engageable*** – Membuat keterlibatan ulang mudah melalui fitur seperti *push notifications*.
- ***Installable*** – Memungkinkan pengguna menambahkan aplikasi ke *homescreen* mereka tanpa harus kesusahan harus melalui toko aplikasi seperti *playstore* dan *appstore*.

- ***Linkable*** – Mudah berbagi aplikasi melalui URL, tidak memerlukan instalasi yang rumit. (Lepage Pete, 2018)

2.2.4 Service Worker

Service worker adalah *script* yang dijalankan oleh browser di latar belakang, terpisah dari halaman web. Dimasa depan, *service worker* mungkin mendukung hal-hal lain seperti sinkronisasi berkala atau *geofencing*. Fitur inti dalam pembahasan ini yaitu untuk mengatasi dan mengelola permintaan jaringan, termasuk mengelola *cache* dari sebuah *responses*. *Service worker* bertindak sebagai perantara browser dan jaringan internet. Sebelum berjalan di jaringan internet, *request* terlebih dahulu akan berjalan melalui *service worker*. Pada kondisi ini, kita sebagai *web developer* dapat mengambil kontrol penuh atas *request*. Begitu pula sebaliknya, ketika response jaringan datang, response tersebut akan melalui *service worker* terlebih dahulu sebelum ke browser. Berikut *life cycle* dari *service worker* :



Gambar 2.2. *Life Cycle Service Worker*

Gambar 2.2 menjelaskan *life cycle service worker* pada instalasi pertama, berikut penjelasannya :

Untuk memasang *service worker* ke situs web, maka harus mendaftarkannya, yang dikerjakan di halaman javascript. Setelah mendaftarkan *service worker*, maka browser akan melakukan instalasi *service worker* di latar belakang. Selama instalasi, bisa dilakukan penyimpanan beberapa file statis dari web. Jika semua file berhasil di *cache*, maka *service worker* akan di *install*. Jika ada file yang gagal diunduh dan disimpan di *cache*, langkah penginstalan akan gagal dan *service worker* tidak akan diaktifkan atau tidak akan di *install*.

Ketika di *install*, langkah selanjutnya yaitu aktifasi dan disini lah kita akan menangani *cache* lama yang sudah tersimpan jika ada.

Setelah langkah aktifasi, *service worker* akan mengontrol semua halaman yang termasuk dalam ruang lingkupnya, meskipun halaman yang mendaftarkan *service worker* untuk pertama kali tidak akan dikontrol sampai halaman dimuat lagi. Setelah *service worker* kembali diaktifkan, ia akan berada dalam satu dari dua kondisi, baik *service worker* akan dihentikan untuk menghemat memori atau akan menangani pengambilan data dan pesan yang terjadi ketika ada yang terjadi dari jaringan atau ada pesan yang dimuat dari sebuah halaman website (Gaunt Matt, 2018).

2.2.5 HTTPS

Hypertext Transfer Protocol (HTTP) adalah protokol yang mengatur komunikasi antara *client* dan *server*. Yang menjadi *client* adalah *web browser*

atau *device* lain yang dapat mengakses, menerima dan menampilkan konten web. Pada umumnya cara komunikasi antara *client* dan *server* adalah *client* melakukan request ke *server*, kemudian *server* mengirimkan respon terhadap *client*. Respon yang dimaksud dapat berupa file HTML yang akan ditampilkan di *browser* ataupun data lain yang di*request* oleh *client*. Semua kegiatan tersebut diatur oleh suatu protocol yaitu *HTTP*.

Sedangkan *Hypertext Transfer Protocol Secure (HTTPS)* adalah versi *secure* dari *HTTP* yang dikembangkan oleh *Netscape Communications Corp*. HTTPS dapat menjamin keamanan data yang ditransmisikan antara client dengan server. Ada 3 aspek yang ditangani oleh *HTTPS*, yaitu

Autentikasi Server, dengan adanya autentikasi *server*, pengguna yakin sepenuhnya bahwa sedang berkomunikasi dengan server yang dituju.

Kerahasiaan Data, data yang ditransmisikan tidak akan bisa dipahami oleh pihak lain, karena data yang ditransmisikan sudah dienkripsi.

Integritas Data, data yang sedang ditransmisikan tidak dapat diubah oleh pihak lain, karena akan divalidasi oleh *message authentication code (MAC). (*Lepage Pete, 2018*).*