Metode Komponisasi Untuk Pembuatan Antar Muka Pengguna Untuk Aplikasi Berbasis Web Berdomain Server Side

Benny Leonard Enrico Panggabean
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Pancasakti
Makassar, Indonesia
blep@unpacti.ac.id

Abstrak

Abstrak Pembangunan sebuah aplikasi khususnya berbasis web dalam sepuluh tahun terakhir mengalami kemajuan yang sangat signifikan baik dari sisi teknologi maupun metodologi. Tetapi menimbulkan masalah dalam kompleksitas pembangunan sebuah aplikasi berbasis web. untuk mengurangi kompleksitas tersebut salah satu metodologi yang digunakan dalam membangun aplikasi berbasis web adalah dengan menggunakan metode komponisasi. Hasil menunjukkan penggunaan metode komponisasi berhasil mengurangi penggunaan memori dan jumlah kode yang dituliskan pada pembangunan sebuah aplikasi berbasis web secara signifikan.

Kata Kunci: *Komponisasi, Web Component, Web Application, PHP, HTML.*

Abstract

Abstract The development of an application, especially web-based in the last ten years has progressed very significantly both in terms of technology and methodology. But it causes problems in the complexity of building a web-based application. to reduce the complexity, one of the methodologies used in building web-based applications is to use the method of componation. The results show that the use of the componation method has significantly reduced memory usage and the amount of code written on the construction of a web-based application.

Keyword: Compounding, Web Component, Web Application, PHP, HTML.

1. Pendahuluan

Saat ini, Pengembangan sebuah aplikasi berbasis web banyak menggunakan kode yang berulang. Untuk mengatasi penggunaan kode yang berulang maka muncul sebuah metode yaitu komponisasi yang bertujuan untuk membangun sebuah aplikasi menggunakan kumpulan komponen dimana salah satu fungsi utamanya adalah enkapsulasi kode secara berulang (Overson & Strimpel, 2015). Dalam aplikasi web yang tidak bersifat monolitik, pengembangannya terbagi atas dua bagian yaitu Server Side Development dan Client Side Development (Orfali & Harkey, 2007).

Server side development merupakan pemrograman web yang pengolahannya dilakukan dalam server yang telah terintegrasi oleh web engine. Dimana peran web engine adalah memproses semua script termasuk kategori client side. Web engine harus diinstal dalam komputer yang terpisah dari web server (Herron, 2018). Client side development merupakan pemrograman web dimana pengolahannya dilakukan dalam

server yang web servernya sudah terintegrasi oleh web engine. Peran web engine adalah memproses semua script yang ada termasuk kategori client side. Web engine harus diinstal dalam komputer dan terpisah dari web server (Gilboa, 2011).

Dalam lima tahun terakhir beberapa framework/library telah diluncurkan untuk membantu dalam mempercepat pengembangan aplikasi berbasis web yang memiliki metode komponisasi yakni keunggulan dalam implementasi source code. Dari sisi Server Side Development ada dua framework yang mempunyai metode komponisasi yaitu Laravel yang menggunakan bahasa pemograman php dan bahasa pemrograman C# (Stauffer, 2019). sedangkan, dari sisi Client Side Development ada tiga yaitu angularJS (Green & Seshadri, 2013) yang dikembangkan oleh google, Vuejs (Patrylo & Milosz, 2017) yang dikembangkan oleh komunitas dan ReactJS (Fedosejev, 2015) yang dikembangkan oleh facebook.

Penelitian ini mengusulkan pemanfaatan konsep komponen untuk pembuatan antar muka pengguna dalam pengembangan sebuah aplikasi berbasis web yang berdomain server side dengan menggunakan Laravel framework dengan Bahasa pemograman PHP

2. Metodologi

2.1. Pemrograman Berorientasi Objek

Pemograman Berorientasi Objek adalah sebuah paradigma pemograman berdasarkan konsep objek, dimana sebuah objek terdiri data dan metode, dimana metode dapat berupa sebuah prosedur maupun sebuah fungsi, dimana objek – objek ini diciptakan untuk berinterkasi satu dengan yang lain. Beberapa Bahasa pemograman yang menggunakan konsep pemograman berorientasi objek adalah java, php, ruby, swift dan scala (Lerdorf et al., 2002), (Prettyman, 2016), (Lopez, 2016), (Cutler & Dickenson, 2020).

Konsep dari OOP sendiri adalah semua pemecahan masalah dibagi ke dalam obyek. Dalam konsep OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai obyek.

1. Object
   Sederhananya, sebuah obyek adalah kumpulan dari variabel dan fungsi yang dibungkus menjadi satu entitas. Entitas tersebut dapat berupa variabel biasa, sebuah obyek diciptakan melalui sebuah kelas atau dengan istilah instance of class. Obyek memiliki 2 elemen utama:
   a. Attributes atau Properties
      Yaitu nilai-nilai yang tersimpan dalam obyek tersebut dan secara langsung maupun tidak langsung menentukan karakteristik dari obyek tersebut.
   b. Method
      Yaitu suatu aksi yang akan dijalankan atau dikerjakan oleh obyek tersebut.
2. Class
   Class dapat didefinisikan sebagai struktur data atau cetak biru dari suatu obyek. Lebih jelasnya adalah sebuah bentuk dasar atau blueprint yang mendefinisikan variabel, method umum pada semua obyek. Obyek sendiri adalah kumpulan variabel dan fungsi yang dihasilkan dari template khusus atau disebut class. Obyek adalah elemen

pada saat run-time yang akan diciptakan, dimanipulasi, dan dibuang/di-destroy ketika eksekusi. Adapun class merupakan definisi statik dari himpunan obyek yang mungkin diciptakan sebagai instantiasi dari class

3. Inheritance
   Inheritance atau dalam bahasa indonesianya disebut sebagai pewarisan adalah suatu cara untuk membuat sebuah kelas yang baru dengan menggunakan kelas lain yang sebelumnya sudah dibuat. Pada hubungan inheritance, sebuah class turunan mewarisi kelas leluhur (parent class). Oleh karena mewarisi, maka semua atribut dan method class dari induk akan dibawa (kecuali yang bersifat private), secara intrinsik menjadi bagian dari class anak. Adapun keuntungan yang didapat dari inheritance menambah fitur baru pada kelas anak dan mengubah atau mengganti fitur yang diwarisi dari kelas parent.

2.2. Laravel

Laravel adalah sebuah Framework PHP dirilis dibawah lisensi MIT dengan kode sumber yang sudah disediakan oleh Github, sama seperti framework-framework yang lain, Laravel dibangun dengan konsep MVC (Model-Controller-View), kemudian Laravel dilengkapi juga command line tool yang bernama "Artisan" yang bisa digunakan untuk packaging bundle dan instalasi bundle melalui command prompt (Bean, 2015). Berikut ini beberapa fitur yang dimiliki oleh framework Laravel

1. Bundles
   Bundles yaitu sebuah fitur dengan system pengemasan modular dan berbagai bundle telah tersedia untuk digunakan dalam aplikasi Anda.
2. Eloquent ORM
   Eloquent ORM merupakan penerapan PHP lanjutan dari pola "active record" menyediakan metode internal untuk mengatasi kendala hubungan antara objek database. Pembangun query Laravel Fluent didukung Eloquent.
3. Application Logic
   Application Logic merupakan bagian dari aplikasi yang dikembangkan, baik menggunakan Controllers maupun sebagai bagian dari deklarasi Route. Sintaks yang digunakan untuk mendefinisikannya mirip dengan yang digunakan oleh framework Sinatra
4. Reverse Routing
   Reverse Routing mendefinisikan hubungan antara link dan route, sehingga jika suatu saat ada perubahan pada route secara otomatis akan tersambung dengan link yang relevan. Ketika link yang dibuat dengan menggunakan nama-nama dari route yang ada, secara otomatis laravel akan membuat URI yang sesuai.
5. Restful Controllers
   Restful Controllers memberikan sebuah option (pilihan) untuk memisahkan logika dalam melayani HTTP GET dan permintaan POST.
6. Class Auto Loading

Class Auto Loading menyediakan otomatis loading untuk class-class PHP, tanpa membutuhkan pemeriksaan manual terhadap jalur masuknya. Fitur ini mencegah loading yang tidak perlu.

7. View Composers
   View Composers adalah kode unit logical yang dapat dijalankan ketika sebuah view di load.
8. IoC Container
   IoC Container memungkinkan untuk objek baru yang dihasilkan dengan mengikuti prinsip control pembalik, dengan pilihan contoh dan referensi dari objek baru sebagai Singletons.
9. Migrations
   Migrations menyediakan versi sistem control untuk skema database, sehingga memungkinkan untuk menghubungkan perubahan adalah basis kode aplikasi dan keperluan yang dibutuhkan dalam merubah tata letak database. Mempermudah dalam penempatan dan memperbarui aplikasi.
10. Unit Testing
    Unit Testing mempunyai peran penting dalam framework Laravel, dimana unit testing ini mempunyai banyak tes untuk mendeteksi dan mencegah regresi. Unit testing dapat dijalankan melalui fitur “artisan command-line”.
11. Automatic Pagination
12. Automatic Pagination menyederhanakan tugas dari penerapan halaman, menggantikan penerapan yang manual dengan metode otomatis yang terintegrasi ke Laravel.

3. Eksperimental

3.1. Komponen

Komponen yang dibuat adalah tiga buah komponen, komponen tersebut adalah komponen pencarian, komponen modal penambahan data, dan komponen perubahan data, komponen perubahan data terdiri atas dua bagian, bagian pertama adalah merubah data, dan bagian kedua adalah konfirmasi penghapusan data. Sebuah komponen terdiri atas dua bagian, bagian pertama adalah bagian deklarasi komponen dan bagian kedua adalah bagian tampilan komponen. dalam implementasi dibuat tiga buah komponen yang akan sering digunakan dalam sebuah applikasi, komponen ini ditulis menggunakan Bahasa pemograman PHP versi 7.4 dan HTML5

Komponen pencarian adalah komponen yang berfungsi sebagai penyaring data dalam sebuah modul aplikasi, penyaringan dapat berupa pencarian dari sebuah kata maupun besarnya data yang akan ditampilkan ke pengguna. Bentuk implementasi komponen yang dibuat dijabarkan dalam koding berikut ini

```php
<?php

declare (strict_types = 1);

namespace App\View\Components\Input;

use Illuminate\View\Component;
```

```
class Search extends Component
{
    public array $selectOptionPaginate;
    public string $path;
    public string $method;
    public string $search;
    public string $identifier;
    public int $selectedPaginate;
    public function __construct(string $path, string $method = 'GET', array $selectOptionPaginate = [], int
$selectedPaginate = 15, string $search = '', string $identifier = '')
    {
        $this->path = $path;
        $this->method = 'GET';
        $this->selectOptionPaginate = [15, 30, 50, 100];
        $this->selectedPaginate = 15;
        $this->search = '';
        $this->identifier = '';
        if (!empty($selectOptionPaginate)) {
            $this->selectOptionPaginate = $selectOptionPaginate;
        }
        if (!empty($method)) {
            $this->method = $method;
        } else {
            $this->method = 'GET';
        }
        if (!empty($selectedPaginate)) {
            $this->selectedPaginate = $selectedPaginate;
        }
        if (!empty($search)) {
            $this->search = $search;
        }
        if (!empty($identifier)) {
            $this->identifier = $identifier;
```

```
    }
  }
  public function render()
  {
    return view('components.input.search');
  }
}
```

Gambar 1. Deklarasi Komponen Pencarian.

```
@if (!empty($identifier))
<form action="{{ route($path, $identifier) }}" method="{{$method}}">
@else
<form action="{{ route($path) }}" method="{{$method}}">
@endif
  @if (strtoupper($method) == 'POST')
    @csrf
  @endif
  <div class="flex flex-wrap -mx-1">
    <div class="w-full my-1 px-1 sm:w-full md:w-1/3 lg:w-1/6 xl:w-1/6">
      <div class="relative">
        <div class="relative">
          <select class="block appearance-none w-full bg-transparent border border-gray-500 text-gray-300 h-10
px-2 px-4 pr-8 rounded-none leading-tight focus:outline-none focus:border-gray-500" name="limit">
            @foreach($selectOptionPaginate as $selectOption)
              <option value="{{$selectOption}}" {{$selectedPaginate == $selectOption ? 'selected':null
}}>{{$selectOption}}</option>
            @endforeach
          </select>
          <div class="pointer-events-none absolute inset-y-0 right-0 flex items-center px-2 text-gray-300">
            <svg class="fill-current h-4 w-4" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"><path
d="M9.293 12.95l.707.707L15.657 8l-1.414-1.414L10 10.828 5.757 6.586 4.343 8z"/></svg>
          </div>
        </div>
      </div>
```

```
    </div>
    <div class="w-full my-1 px-1 sm:w-full md:w-2/3 lg:w-5/6 xl:w-5/6">
      <div class="flex flex-wrap items-stretch w-full mb-4 relative">
        <input
          name="search"
          type="text"
          class="flex-shrink flex-grow flex-auto leading-normal w-px flex-1 border h-10 px-2 relative text-sm
border-r-0 bg-transparent border-gray-500 text-gray-300"
          placeholder="Search"
          value="{{ $search }}"
        />
        <div class="flex -mr-px">
          <button
            type="submit"
            class="px-2 flex items-center whitespace-no-wrap text-sm leading-normal border border-l-0 bg-
transparent border-gray-500 text-gray-300"
          >
            <i class="mdi mdi-18px mdi-magnify"></i>
          </button>
        </div>
      </div>
    </div>
  </div>
</form>
```

Gambar 2. Deklarasi Tampilan Komponen Pencarian.

Komponen penambahan data adalah sebuah komponen yang berfungsi untuk menambahkan data pada sebuah modul

```
<?php
declare (strict_types = 1);
namespace App\View\Components\Modal;
use Illuminate\View\Component;
use Illuminate\Support\ViewErrorBag;
```

```
class ReferenceCreate extends Component
{
    public string $path;
    public string $isWithAlphabetCode;
    public array $data;
    public ViewErrorBag $errors;
    public function __construct(string $path, array $data = [], ViewErrorBag $errors, string $isWithAlphabetCode =
'true')
    {
        $this->path = $path;
        $this->isWithAlphabetCode = $isWithAlphabetCode;
        $this->data = $data;
        $this->errors = $errors;
        $this->data['action'] = 'create';
    }
    public function render()
    {
        return view('components.modal.reference-create');
    }
}
```

Gambar 3. Deklarasi Komponen Penambahan Data.

```
button id="btn-open-create-modal" class="btn-open-modal bg-transparent text-gray-300 p-2 mr-2 border border-
gray-500 hover:text-green-500 hover:border-green-500" data-window-modal="create-modal">
    <i class="mdi mdi-18px mdi-plus"></i>
</button>
<div id="create-modal" class="modal-window fixed bottom-0 inset-x-0 px-4 pb-4 sm:inset-0 sm:flex sm:items-center
sm:justify-center z-20 {{($errors->any() && old('action') === 'create') ? 'visible':'invisible'}}">
    <div class="fixed inset-0 transition-opacity">
        <div class="absolute inset-0 bg-gray-500 opacity-75 blur"></div>
    </div>
    <div class="bg-gray-800 rounded-none overflow-hidden shadow-xl transform transition-all sm:max-w-lg sm:w-
full">
        <div class="bg-gray-800 px-4 pt-5 pb-4 sm:p-6 sm:pb-4 rounded-none">
```

```
        <div class="sm:flex sm:items-start">
          <div class="mx-auto flex-shrink-0 flex items-center justify-center h-12 w-12 rounded-full bg-red-100
sm:mx-0 sm:h-10 sm:w-10">
            <i class="mdi mdi-24px mdi-plus text-green-500"></i>
          </div>
          <div class="mt-3 text-center sm:mt-0 sm:ml-4 sm:text-left">
            <h3 class="text-lg mt-2 leading-6 font-medium text-gray-300">
              Add New Data
            </h3>
          </div>
        </div>
      </div>
      <form method="POST" action="{{route($path)}}">
        @csrf
        <input type="hidden" name="action" value="create">
        <div class="mt-2 mx-6">
          <label class="block text-gray-300 text-sm font-bold mb-2" for="code">
            Code
          </label>
          <input class="bg-transparent shadow appearance-none border {{$errors->has('code') ? 'border-red-
500':'border-gray-500'}} rounded-none w-full py-2 px-3 text-gray-300 mb-3 leading-tight focus:outline-none
focus:shadow-outline" id="code" name="code" type="text" placeholder="please input code"
value="{{old('code')}}">
          @if($errors->has('code') && old('action') === 'create')
            <p class="text-red-500 text-xs italic">{{$errors->first('code')}}</p>
          @endif
        </div>
        @if($isWithAlphabetCode == 'true')
          <div class="mt-2 mx-6">
            <label class="block text-gray-300 text-sm font-bold mb-2" for="alphabet_code">
              Alphabet Code
            </label>
```

```
            <input class="bg-transparent shadow appearance-none border {{$errors->has('alphabet_code') ?
'border-red-500':'border-gray-500'}} rounded-none w-full py-2 px-3 text-gray-300 mb-3 leading-tight focus:outline-
none focus:shadow-outline" id="alphabet_code" name="alphabet_code" type="text" placeholder="please input
alphabet code" value="{{old('alphabet_code')}}">
            @if($errors->has('alphabet_code') && old('action') === 'create')
              <p class="text-red-500 text-xs italic">{{$errors->first('alphabet_code')}}</p>
            @endif
          </div>
        @endif
        <div class="mt-2 mx-6">
          <label class="block text-gray-300 text-sm font-bold mb-2" for="name">
            Name
          </label>
          <input class="bg-transparent shadow appearance-none border {{$errors->has('name') ? 'border-red-
500':'border-gray-500'}} rounded-none w-full py-2 px-3 text-gray-300 mb-3 leading-tight focus:outline-none
focus:shadow-outline" id="name" name="name" type="text" placeholder="please input name"
value="{{old('name')}}">
          @if($errors->has('name') && old('action') === 'create')
            <p class="text-red-500 text-xs italic">{{$errors->first('name')}}</p>
          @endif
        </div>
        <div class="bg-gray-50 px-4 py-3 sm:px-6 sm:flex sm:flex-row-reverse">
          <span class="flex w-full rounded-md shadow-sm sm:ml-3 sm:w-auto">
            <button id="btn-close-create-modal" type="button" class="btn-close-modal inline-flex justify-center w-
full rounded-none border border-gray-500 p-2 bg-transparent text-base leading-6 font-medium text-gray-500
shadow-sm hover:bg-red-500 focus:outline-none focus:border-white hover:text-white focus:shadow-outline-red
transition ease-in-out duration-150 sm:text-sm sm:leading-5" data-window-modal="create-modal">
              <i class="mdi mdi-18px mdi-close"></i>
            </button>
          </span>
          <span class="flex w-full rounded-md shadow-sm sm:ml-3 sm:w-auto">
            <button type="submit" class="inline-flex justify-center w-full rounded-none border border-gray-500 p-2
bg-transparent text-base leading-6 font-medium text-gray-500 shadow-sm hover:bg-green-500 focus:outline-none
```

```
focus:border-white hover:text-white focus:shadow-outline-red transition ease-in-out duration-150 sm:text-sm
sm:leading-5">
                    <i class="mdi mdi-18px mdi-check"></i>
                </button>
            </span>
        </div>
    </form>
  </div>
</div>
@push('scripts')
<script>
  document.addEventListener("DOMContentLoaded", function() {
    @if($errors->any() && old('action') === 'create')
    if(!$('body').hasClass('overflow-hidden')) {
        $('body').toggleClass('overflow-hidden');
    }
    @endif
    $('#btn-open-create-modal').on('click', function() {
        let windowModal = $('#btn-open-create-modal').data('window-modal');
        $('body').toggleClass('overflow-hidden');
        $('#'+windowModal).toggleClass('invisible');
    });
    $('#btn-close-create-modal').on('click', function() {
        let windowModal = $('#btn-close-create-modal').data('window-modal');
        $('body').toggleClass('overflow-hidden');
        $('#'+windowModal).toggleClass('invisible');
    });
  });
</script>
@endpush
```

Gambar 4. Deklarasi Tampilan Komponen Penambahan Data.

Komponen Perubahan data adalah komponen yang berfungsi melakukan pengubahan pada data atau melakukan fungsi penghapusan sebuah data

```php
<?php

declare (strict_types = 1);

namespace App\View\Components\Modal;

use Illuminate\View\Component;

use Illuminate\Support\ViewErrorBag;

class ReferenceAction extends Component

{

    public string $identifier;

    public string $editPath;

    public string $deletePath;

    public array $modelData;

    public string $isWithAlphabetCode;

    public string $showPath;

    public ViewErrorBag $errors;

    public function __construct(string $identifier, string $editPath, string $deletePath, array $modelData,

ViewErrorBag $errors, string $isWithAlphabetCode = 'false', string $showPath = '')

    {

        $this->identifier = $identifier;

        $this->deletePath = $deletePath;

        $this->editPath = $editPath;

        $this->modelData = $modelData;

        $this->isWithAlphabetCode = $isWithAlphabetCode;

        $this->showPath = $showPath;

        $this->modelData['action'] = 'modified';

        $this->errors = $errors;

    }

    public function render()

    {

        return view('components.modal.reference-action');

    }

}
```

Gambar 5. Deklarasi Komponen Perubahan Data.

Gambar 1 sampai gambar 5 merupakan implementasi sebelum penggunaan komponen sebuah modul yang memiliki 397 kode line baris. Setelah proses komponisasi yaitu dengan mengubah kode diatas dengan mengganti bagian – bagian coding module menjadi sebuah komponen maka hasilnya dapat dilihat seperti pada gambar 6 berikut ini

```
@extends('layouts.administrator')
@php
    $indexPath = 'web.person.reference.identification_type.index';
    $storePath = 'web.person.reference.identification_type.store';
    $updatePath = 'web.person.reference.identification_type.update';
    $destroyPath = 'web.person.reference.identification_type.destroy';
    if($errors->any() && !empty(old('action')) && old('action') === 'create' ) {
        notify()->error("Fail To Create", "Error", "topLeft");
    } else if ($errors->any() && !empty(old('action')) && old('action') === 'modified' ) {
        notify()->error("Fail To Update", "Error", "topLeft");
    }
@endphp
@section('content')
<div class="clear-both"></div>

<div class="mx-4 pt-24 mb-16 text-gray-100 min-h-full">
    <div class="flex justify-end">
        <x-modal.reference-create :path="$storePath" :errors="$errors" />
        <x-modal.manual identifier="manual-information" />
    </div>
    <br />
    <x-input.search :path="$indexPath" method="GET" />
    <table class="table-auto w-full">
        <thead>
            <tr>
                <th class="border border-gray-500 px-4 py-2 text-gray-300">Code</th>
                <th class="border border-gray-500 px-4 py-2 text-gray-300">Alphabet Code</th>
                <th class="border border-gray-500 px-4 py-2 text-gray-300">Name</th>
                <th class="border border-gray-500 px-4 py-2 text-gray-300" width=30>Act</th>
            </tr>
```

```
    </thead>
    <tbody>
      @foreach($models as $model)
        <tr>
          <td class="border border-gray-500 px-4 py-2 text-gray-300">{{$model->code}}</td>
          <td class="border border-gray-500 px-4 py-2 text-gray-300">{{$model->alphabet_code}}</td>
          <td class="border border-gray-500 px-4 py-2 text-gray-300">{{$model->name}}</td>
          <td class="border border-gray-500 px-4 py-2" width=30>
            <x-modal.reference-action :identifier="$model->id" is-with-alphabet-code="true" :edit-
path="$updatePath" :delete-path="$destroyPath" :errors="$errors" :model-data="$model->toArray()" />
          </td>
        </tr>
      @endforeach
    </tbody>
  </table>
  <div class="clear-both"></div>
  <br />
  {{ $models->links() }}
  <div class="clear-both"></div>
  <br />
</div>
@endsection
```

Gambar 6. Kode Modul Aplikasi menggunakan metode komponisasi.

4. Hasil

Sebelum proses komponisasi dilakukan dari lima puluh kali percobaan pengaksesan halaman diperoleh data yang terdiri dari tiga variabel, variabel tersebut adalah ResponseEnd, variabel ini berfungsi untuk mendeteksi jumlah waktu yang dibutuhkan ketika sebuah request dari server ke client selesai dilaksanakan, variabel berikutnya adalah variabel DOMContentLoadedEventEnd, variabel ini berfungsi untuk mendeteksi jumlah waktu yang dibutuhkan oleh peramban untuk merender sebuah document object model. Variabel terakhir adalah LoadEventEnd , variabel ini berfungsi untuk mendeteksi jumlah waktu yang dibutuhkan oleh peramban untuk mempersiapkan document object model sebelum di render. Ketiga variabel ini menggunakan satuan milidetik.

Tabel 1. Hasil pengaksesan halaman

Trial	ResponseEnd	DOMContentLoadedEventEnd	LoadEventEnd
1	572	597	1031
2	289	309	487
3	364	381	386
4	279	294	297
5	331	348	352
6	283	301	305
7	268	284	288
8	267	283	288
9	266	283	286
10	268	284	288
11	280	297	302
12	285	301	303
13	451	475	479
14	647	666	672
15	392	414	418
16	368	389	393
17	318	338	345
18	297	313	318
19	290	308	312
20	281	297	301
21	283	299	303
22	299	316	320
23	285	302	309
24	297	314	318
25	282	298	302
26	280	296	300
27	297	327	327
28	282	299	303
29	276	294	299
30	285	302	306
31	380	399	404
32	409	427	431
33	368	388	393
34	273	291	297
35	332	352	357
36	346	363	368
37	288	304	308
38	292	325	326
39	283	301	305
40	298	314	319
41	286	301	305
42	331	350	355
43	379	400	405
44	311	326	330
45	322	341	346

Trial	ResponseEnd	DOMContentLoadedEventEnd	LoadEventEnd
46	289	306	309
47	270	287	290
48	319	340	345
49	299	318	324
50	294	314	319

Setelah mengganti sebagian kode dengan komponen jumlah baris kode yang dituliskan menjadi 51 baris kode, perubahan jumlah baris kode dari total 397 baris ke 51 baris menghasilkan penghematan penulisan kode sebuah modul sebesar 87.15%. Penghematan ini sangat berpengaruh bagi kinerja developer, selain mengurangi kesalahan penulisan kode, juga mempercepat pengerjaan sebuah modul. Hal yang menjadi rintangan terberat dalam membuat komponen salah satunya sejauh mana fungsi yang ingin dihasilkan oleh sebuah komponen. Setelah menghitung nilai rata – rata dari ketiga variabel baik menggunakan metode komponisasi dan tanpa metode komponisasi diperoleh hasil sebagai berikut

Tabel 2. Perbandingan metode.

Metode	ResponseEnd	DOMContentLoadedEvent End	LoadEventEnd
Komponisasi	320,62	339,12	355,48
Tanpa Komponisasi	290,34	306,36	316,86
Selisih Waktu	30,28	32,76	38,62

Tabel diatas menunjukkan bahwa perbedaan nilai waktu yang dibutuhkan oleh peramban yang ditunjukkan oleh ketiga variabel tersebut tidak signifikan.

5. Kesimpulan

Pembuatan sebuah komponen dalam pembangunan sebuah aplikasi, bergantung dari struktur dari aplikasi tersebut, tidak harus semua bagian dari kode modul sebuah aplikasi di komponisasi, semuanya harus dilihat dari tujuan akhir penggunaan modul tersebut, bahwa perubahan sebahagian kode modul applikasi menjadi sebuah komponen bisa membawa pengaruh yang signifikan pada pembangunan sebuah aplikasi, terutama pada aplikasi yang berskala besar, pengaruh positif yang dihasilkan dengan menggunakan pendekatan pembangunan aplikasi dengan komponisasi adalah jumlah kode yang harus dituliskan pada sebuah modul applikasi menjadi jauh lebih singkat, hal ini berpengaruh pada kinerja developer dalam membangun sebuah modul aplikasi

Daftar Pustaka

Overson, J., & Strimpel, J. (2015). Developing Web Components: *UI from jQuery to Polymer*. " O'Reilly Media, Inc.".

Orfali, R., & Harkey, D. (2007). *CLIENT/SERVER PROGRAMMING WITH JAVA AND CORBA, (With CD)*. John Wiley & Sons.

Herron, D. (2018). *Node. js Web Development: Server-side development with Node 10 made easy*. Packt Publishing Ltd.

Gilboa, Y. (2011). *U.S. Patent No. 7,971,194*. Washington, DC: U.S. Patent and Trademark Office.

Stauffer, M. (2019). Laravel: Up & Running: *A Framework for Building Modern PHP Apps*. O'Reilly Media.

Green, B., & Seshadri, S. (2013). *AngularJS*. " O'Reilly Media, Inc.".

Patrylo, N., & Milosz, M. (2017). Comparison of AngularJS and VueJS frameworks efficiency. *Journal of Computer Sciences Institute, 5*, 204-207.

Fedosejev, A. (2015). *React. js essentials*. Packt Publishing Ltd.

Lerdorf, R., Tatroe, K., Kaehms, B., & McGredy, R. (2002). Programming Php. " O'Reilly Media, Inc.".

Prettyman, S. (2016). *Learn PHP 7*. Apress.

Lopez, A. (2016). Learning Php 7. Packt Publishing Ltd.

Cutler, J., & Dickenson, M. (2020). Object-Oriented Programming. In Computational Frameworks for Political and Social Research with Python (pp. 33-48). Springer, Cham.

Bean, M. (2015). *Laravel 5 essentials*. Packt Publishing Ltd.