Implementasi *Component-Based Development* Untuk Pengembangan *Front-End* Pada *Website* Feelsbox (Studi Kasus: Fitur Kursus *Online* FeelsQuest)

1st Asyrafbilal Fadhila Bhinar Jaya
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
asyrafbilalrpltelu@student.telkomuniversity.ac.id

2nd Dana Sulistyo Kusumo
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
danakusumo@telkomuniversity.ac.id

3rd Nungki Selviandro
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
nselviandro@telkomuniversity.ac.id

***Abstrak* — Masalah yang sering dihadapi oleh *front-end web* developer adalah penulisan kode program dengan kegunaan yang sama secara berulang-ulang sehingga mempersulit proses pemeliharaan. Pada penelitian ini dilakukan pengembangan *front-end website* fitur FeelsQuest dengan menggunakan *Component-Based Development*(CBD). Penerapan CBD ini bertujuan untuk membuat tampilan *website* dikembangkan secara komponen tampilan kecil yang dapat dengan mudah dipelihara. Penelitian ini menggunakan *framework* Laravel yang terfokus pada modul View dan fitur Laravel Blade Templates yang dapat membuat komponen tampilan *website* untuk pengimplementasian CBD. Pada pengembangan komponen akan menerapkan prinsip SOLID untuk menghasilkan kode program yang mudah dipelihara. Hasil kode program antarmuka *website* diuji menggunakan *tools* PHPMetrics untuk mendapatkan matriks *Maintainability* Index(MI) yang menjadi ukuran apakah kode program dapat dengan mudah dimaintain atau tidak. Nilai rata-rata matriks MI yang didapat adalah 110.20, yakni dapat dikatakan bagus karena bernilai 85 keatas, artinya kode program dapat dengan mudah dialakukan perubahan, perbaikan dan penambahan tampilan pada pengembangan selanjutanya. Hasil penelitian ini menunjukkan bahwa pengembangan *front-end website* fitur FeelsQuest dengan mengimplementasikan CBD dan penerapan prinsip SOLID menggunakan Laravel dapat mempermudah proses pemeliharaan.**

Kata kunci*— *Front-End*, Laravel, *Component-Based Development*, *Maintainability

I. PENDAHULUAN

Menurut Riset Kesehatan Dasar (Riskesdas) 2018, lebih dari 19 juta orang yang berusia di atas 15 tahun mengalami gangguan mental emosional, dan lebih dari 12 juta orang yang berusia di atas 15 tahun mengalami depresi [1]. Angka yang tinggi ini menegaskan perlunya langkah-langkah preventif yang efektif untuk mengelola dan mencegah kondisi kesehatan mental yang lebih serius. Feelsbox adalah *startup* berbasis digital yang menekankan pentingnya kesehatan mental dengan menawarkan solusi inovatif melalui berbagai layanannya. Saat ini Feelsbox mengembangkan fitur FeelsQuest yaitu kursus *online* kesehatal mental. Fitur FeelsQuest sebagai langkah previntif dengan memberikan materi tentang gejala awal masalah kesehatan mental serta teknik pengelolaan yang diperlukan. Dengan begitu bisa membantu individu mengidentifikasi dan mengelola masalah kesehatan mental sebelum berkembang menjadi kondisi yang lebih serius. Kursus *online* diadopsi dan diimplementasikan karena merupakan metode yang relatif cepat untuk mendistribusikan bahan ajar, serta materi kursus *online* dapat diubah dengan cepat. [2].

Antarmuka tampilan dari *website* adalah hal yang paling esensial dalam pengembangan *website* kursus *online*. Ini adalah titik kontak antara pengguna dan sistem untuk menentukan kegunaan dan efektivitas sistem [2]. Dalam pengembangan *Front-End website* masalah yang sering dihadapi pengembang adalah sulitnya pemeliharaan karena penulisan ulang kode program karena jika ada tampilan yang ingin ditambahkan atau diubah maka pengembang harus mengubah semua kode program dengan menuliskan ulang kode program, yang dimana hal tersebut memperumit pengembangan.

Pada pengembangan ini dibutuhkan pendekatan pengembangan yang dapat membuat komponen untuk memudahkan pemeliharaan kode program. Pendekatan yang dapat digunakan untuk menangani masalah ini adalah *Component-Based Development* (CBD) dimana pendekatan tersebut menawarkan pengembangan perangkat lunak yang dibangun menggunakan komponen-komponen yang dapat digunakan kembali. Komponen yang dikembangkan menerapkan prinsip SOLID agar mendukung kode program yang mudah dipelihara. Ini adalah solusi untuk menghilangkan kelemahan pengembangan perangkat lunak tradisional dengan meningkatkan produktivitas, pemeliharaan, dan mengurangi waktu dan biaya pengembangan [3]. *Framework* yang digunakan untuk menerapkan pendekatan CBD adalah Laravel. Laravel memiliki fitur Laravel Blade Templates yang memungkinkan untuk membuat PHP dan HTML kustom baru yang dapat digunakan kembali dan dienkapsulasi. Laravel MVC memberikan keunggulan dalam struktur terorganisir menjadi Model, View, dan Controller, memungkinkan pengembangan

paralel, kode yang mudah dipelihara, reusabilitas kode, serta skalabilitas yang jelas dalam pengembangan aplikasi [4]. Sedangkan kelemahan Laravel pada kinerja yang lambat dibanding *framework* modern lain dan ketergantungannya pada Composer.

Untuk memastikan bahwa CBD dapat menyelesaikan permasalahan sulitnya proses pemeliharaan, maka dilakukan pengujian *maintainability*. Pengujian dilakukan menggunakan PHPMetric untuk mendapatkan nilai *Maintainability Index*(MI) sebagai acuan apakah *maintainability* dari pengembangan ini sudah baik yaitu diatas 85. Dengan nilai yang baik artinya pengembang dapat dengan mudah memperbaiki, merubah dan menambahkan tampilan baru pada pengembangan selanjutnya.

Tujuan dari penelitian ini adalah untuk mengimplementasikan CBD dan prinsip SOLID menggunakan *framework* Laravel pada pengembangan *front-end website* fitur FeelsQuest. Selain itu, juga menguji *maintainability* dari kode program antarmuka *website* yang dikembangkan dengan *tools* PHPMetric.

II. KAJIAN TEORI

A. *Front-End*

Front-End web development adalah pengembangan graphical user interface (GUI) dari sebuah *website* atau bisa dibilang antarmuka, yang menggunakan HTML, CSS, dan JavaScript atau juga PHP untuk memungkinkan pengguna melihat dan berinteraksi dengan situs *web*. Ini mencakup semua elemen yang terlihat dan dirasakan oleh pengguna, seperti tata letak, warna, teks, gambar, tombol, formulir, dan elemen-elemen visual lainnya. *Front-End* juga mencakup logika dan fungsionalitas yang memungkinkan pengguna untuk berinteraksi dengan aplikasi atau *website* tersebut.

Front-End memiliki peran penting dalam menciptakan pengalaman pengguna yang baik, karena merupakan bagian yang langsung berhubungan dengan pengguna [5]. Sebuah *Front-End* yang baik harus memiliki tampilan yang menarik, mudah dinavigasi, responsif, dan mudah digunakan oleh pengguna. Selain itu, Front-end juga harus memastikan bahwa aplikasi atau *website* dapat diakses dengan baik di berbagai perangkat dan *browser*.

B. Laravel

Laravel adalah sebuah *framework* berbasis bahasa pemrograman PHP yang digunakan untuk membangun aplikasi *web*. Laravel adalah satu-satunya *framework* yang memungkinkan penggunaan PHP secara maksimal dalam pengembangan *website* [6]. Laravel fokus pada kesederhanaan, kejelasan, dan kesederhanaan dalam penulisan kode dan tampilan, serta menghasilkan fungsionalitas aplikasi *web* yang bekerja sebagaimana mestinya.

Salah satu keunggulan utama dari Laravel adalah kemudahan penggunaan dan dokumentasi yang lengkap, sehingga memungkinkan pengembang untuk membangun aplikasi *web* dengan cepat dan efisien. Selain itu, Laravel juga mendukung konsep pengembangan berbasis MVC (Model-View-Controller) yang memisahkan logika aplikasi, tampilan, dan data [5]. Laravel MVC memberikan keunggulan dalam struktur terorganisir menjadi Model, View, dan Controller, memungkinkan pengembangan paralel, kode yang mudah dipelihara, reusabilitas kode, serta skalabilitas yang jelas dalam pengembangan aplikasi [4].

Laravel memiliki fitur Laravel Blade Templates yaitu bagian dari template blade yang memungkinkan pengembang membuat PHP dan HTML kustom baru yang dapat digunakan kembali dan dienkapsulasi [7]. Misalkan ingin membuat navbar untuk digunakan di banyak halaman aplikasi. Dengan begitu akan memudahkan pengembang untuk menggunakan kembali komponen sesuai kebutuhan. GAMBAR 1 menjelaskan penggunaan komponen navbar yang dipanggil pada suatu halaman.

```
<html>
  <body>
    <x-navbar/>
  </body>
</html>
```

GAMBAR 1
(Kode penggunaan Laravel Blade Component)

Dengan fitur dari Laravel ini, penelitian akan dilakukan dengan menggunakan fitur tersebut untuk menerapkan pendekatan CBD dalam pengembangan *Front-End* FeelsQuest.

C. Bootstrap

Bootstrap adalah *framework web development* yang gratis dan *open-source*, dirancang untuk mempercepat pengembangan *web* yang responsif dan berfokus pada perangkat mobile [8]. Salah satu keunggulan utama dari Bootstrap adalah kemampuannya dalam menyediakan *grid system* yang fleksibel, yang memungkinkan pengguna untuk merancang tata letak halaman *web* yang responsif dengan mudah [9]. Bootstrap juga menyediakan gaya *default* CSS yang dapat diterapkan ke elemen-elemen HTML dengan cepat, membantu dalam proses pengembangan yang efisien.

Dalam pengimplementasiannya pada Laravel, Bootstrap akan dipanggil dalam bentuk *tag link* yang berisi *link* CDN (*Content Delivery Network*) Bootstrap. Dengan begitu Bootstrap dapat digunakan pada atribut *class* dalam *tag* HTML. Berikut adalah GAMBAR 2 yang mencontohkan penggunaan Bootstrap.

Primary link Secondary link

```
<a href="#" class="link-primary">Primary link</a>
<a href="#" class="link-secondary">Secondary link</a>
```

GAMBAR 2
(Kode penggunaan Bootstrap)

Framework ini sangat dibutuhkan untuk pengembangan *Front-End* FeelsQuest karena akan sangat membantu dalam memperindah tampilan antarmuka atau komponen FeelsQuest.

D. *Component-Based Development*

Component-Based Development (CBD) adalah sebuah sistem yang berorientasi pada integrasi yang fokus pada penyusunan komponen-komponen individual untuk membangun sebuah sistem perangkat lunak [10]. CBD adalah pendekatan dalam pengembangan perangkat lunak di mana sistem dibangun menggunakan komponen-komponen yang dapat digunakan kembali [11]. Komponen ini

berkomunikasi satu sama lain melalui tampilan yang digunakan untuk menyediakan layanan. Kelebihan dari pengembangan berbasis komponen adalah bahwa pendekatan ini memungkinkan untuk pengembangan yang lebih sederhana dan dapat menyesuaikan dengan keperluan [10].

Pengembangan *Front-End* menggunakan CBD tidak ada bedanya dengan pengembangan perangkat lunak secara keseluruhan. Dalam penelitian ini, tujuan dari pendekatan CBD adalah agar komponen-komponen *Front-End* dapat digunakan kembali dalam berbagai keperluan dengan harapan dapat memudahkan pemeliharaan. Tahapan penerapan CBD yang dilakukan meliputi [12]:

1. Analisis kebutuhan sistem dan pemilihan komponen yang sesuai dengan kebutuhan tersebut.
2. Desain arsitektur sistem dengan mempertimbangkan lapisan aplikasi atas, komponen bisnis, middleware, dan komponen dasar.
3. Pengembangan komponen-komponen perangkat lunak yang akan digunakan dalam sistem.
4. Integrasi komponen-komponen tersebut untuk membentuk sistem aplikasi yang utuh.
5. Evaluasi arsitektur sistem, pengukuran efisiensi penggunaan sumber daya, dan pengujian sistem untuk memastikan kinerja, keamanan, dan kehandalan sistem.

Mengacu dari tahapan tersebut, tahap pertama tidak termasuk lingkup dari penelitian ini. Komponen yang akan dikembangkan adalah komponen yang telah didesain oleh role UI/UX. Dengan begitu tahapan penerapan CBD dapat saya simpulkan menjadi 4 tahapan seperti berikut:

1. Analisis

Tahapan ini mengacu pada tahap pertama tahapan penerapan CBD. Pada tahap ini dilakukan analisis desain UI untuk menentukan komponen apa saja yang akan dikembangkan menggunakan SOLID *Principles*.

2. Rancangan

Tahapan ini mengacu pada tahap kedua tahapan penerapan CBD. Pada tahap ini membuat rancangan untuk dapat dilakukan pengimplementasian CBD pada Laravel View.

3. Implementasi

Tahapan ini mengacu pada tahap ketiga dan keempat tahapan penerapan CBD. Pada tahap ini dilakukan pengembangan komponen dan pengintegrasian untuk membentuk tampilan yang utuh dengan mengimplementasikan CBD.

4. Evaluasi

Tahap ini mengacu pada tahap kelima tahapan penerapan CBD. Pada tahap ini evaluasi dilakukan dengan pengujian *Maintainability*.

E. SOLID *Principles*

SOLID *Principles* adalah lima prinsip dasar yang membantu menciptakan arsitektur perangkat lunak yang baik [13]. Nama SOLID adalah singkatan dari kelima prinsip yang ada, berikut adalah penjelasan kelima prinsip tersebut.

1. *Single Responsibility Principle* (SRP)

SRP menyatakan bahwa setiap komponen harus memiliki satu tanggung jawab yang terdefinisi dengan baik. Hal ini memastikan bahwa komponen fokus pada satu fungsi spesifik dan tidak terbebani dengan tugas lain yang tidak relevan. Komponen yang mengikuti SRP lebih mudah dipahami, diubah, dan digunakan kembali dalam konteks yang berbeda.

2. *Open-Closed Principle* (OCP)

OCP menyatakan bahwa komponen harus terbuka untuk ekstensi tetapi tertutup untuk modifikasi. Artinya, fungsionalitas baru dapat ditambahkan ke komponen tanpa mengubah kode yang sudah ada. Hal ini memungkinkan komponen untuk beradaptasi dengan perubahan tanpa melanggar stabilitas dan reusabilitasnya.

3. *Liskov Substitution Principle* (LSP)

LSP menyatakan bahwa subkelas harus dapat menggantikan kelas induknya tanpa menyebabkan efek samping yang tidak terduga. Hal ini penting untuk memastikan kompatibilitas antara komponen dan memungkinkan mereka untuk digunakan secara bergantian dalam hierarki kelas.

4. *Interface Segregation Principle* (ISP)

ISP menyatakan bahwa antarmuka tidak boleh terlalu besar dan kompleks. Sebaiknya dipecah menjadi antarmuka yang lebih kecil dan spesifik untuk melayani kebutuhan klien yang berbeda. Hal ini meningkatkan modularitas dan mempermudah penggunaan kembali komponen dalam berbagai skenario.

5. *Dependency Inversion Principle* (DIP)

DIP menyatakan bahwa modul tingkat tinggi tidak boleh bergantung pada modul tingkat rendah, keduanya harus bergantung pada abstraksi. Abstraksi dapat diwujudkan dengan menggunakan antarmuka dan kelas abstrak, yang memungkinkan komponen untuk dihubungkan secara longgar dan diubah tanpa memengaruhi satu sama lain.

Menggunakan SOLID *Principles* dapat membangun perangkat lunak yang efisien, digunakan kembali, berkelanjutan dan dapat dipelihara untuk kebutuhan jangka panjang [13]. Dengan mengikuti SOLID *Principles*, pengkodean komponen dengan CBD menggunakan Laravel dapat mempermudah proses pemeliharaan.

F. *Maintainability*

Maintainability adalah kemudahan di mana sistem atau komponen perangkat lunak dapat dimodifikasi atau diadaptasi terhadap lingkungan yang berubah [14]. *Maintainability* menilai seberapa mudah suatu sistem diperbaiki ketika mengalami kerusakan. Terdapat banyak metrik atau metode yang dapat digunakan untuk mengukur tingkat *Maintainability* perangkat lunak, salah satunya adalah *Maintainability Index*(MI) [15]. Semakin tinggi nilai *Maintainability Index* menunjukkan bahwa kode program memiliki tingkat pemeliharaan yang baik, sehingga kode tersebut lebih mudah dipahami dan dirawat, dimana ini mempermudah dalam menemukan dan memperbaiki *bug*, serta menambahkan fungsionalitas baru. [15].

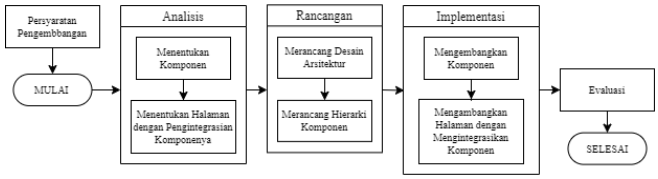
TABEL 1
(Klasifikasi Nilai *Maintainability Index*)

Nilai *Maintainability Index* (MI)	Klasifikasi
MI > 85	*Good maintainability*
65 < MI ≤ 85	*Moderate maintainability*
MI ≤ 65	*Difficult to maintain*

TABEL 1 ini mengidikasi nilai MI dan Klasifikasinya. Kalkulasi nilai MI akan membutuhkan waktu yang lama dan

usaha yang besar jika dilakukan dengan cara manual. Oleh karena itu, penelitian ini menggunakan *tools* untuk mengukur nilai *Maintainability Index* secara otomatis menggunakan PHPMetric.

III. METODE



GAMBAR 3
(Alur Proses Penelitian)

GAMBAR 3 menggambarkan alur proses penelitian atau pengembangan *front-end* fitur FeelsQuest dengan mengimplementasikan CBD. Terdapat empat proses pada penelitian ini, dimulai dari analisis, rancangan, implementasi dan evaluasi. Dalam iterasi *Scrum*, proses analisis dilakukan pada *Sprint* 4, rancangan pada *Sprint* 5, implementasi pada *Sprint* 6-10, dan evaluasi pada *Sprint* 11. Setiap fase iteratif ini memastikan bahwa pengembangan komponen dilakukan secara bertahap dan berkelanjutan, dengan implementasi yang dapat disesuaikan dan diperbaiki berdasarkan umpan balik selama iterasi. Proses ini mendukung fleksibilitas dan responsivitas terhadap perubahan kebutuhan selama siklus pengembangan, sementara evaluasi akhir memastikan bahwa hasil akhir memenuhi standar kualitas dan fungsionalitas yang diinginkan.

A. Persyaratan Pengembangan

Persyaratan pengembangan diperlukan karena sebagai dasar dari pengembangan *front-end* fitur FeelsQuest. Pada pengembangan ini persyaratan berisi *Functional Requirement*, *Non-Functional Requirement* dan rancangan UI yang dirancang oleh UI/UX *designer*. *Functional Requirement* merupakan sekumpulan fitur atau fungsionalitas yang dapat dilakukan oleh aplikasi. Sedangkan *Non-Functional Requirement* merupakan sekumpulan kebutuhan dari suatu sistem yang menentukan perilaku yang harus dipenuhi oleh aplikasi. Persyaratan ini dibuat oleh Tim TA Capstone Feelsbox dengan berdiskusi dengan *stakeholder* Feelsbox. Tujuan persyaratan pengembangan ini adalah menjelaskan terkait dengan apa yang diharapkan dari aplikasi yang sedang dibangun dan bagaimana aplikasi tersebut dapat berinteraksi dengan pengguna.

Functional Requirement menjadi dasar dalam proses pengembangan *front-end* fitur FeelsQuest agar menyediakan tampilan sesuai dengan persyaratan. Sedangkan *Non-Functional Requirement* nomor 6 (NFR06) atau *Maintainability* adalah persyaratan pengembangan *front-end* fitur FeelsQuest yang menjadi dasar perumusam pengujian *Maintainability*.

B. Analisis

Analisis didasarkan dari persyaratan pengembangan pada poin sebelumnya. Analisis dilakukan bersama UI/UX *designer* dan *stakeholder*. Pada tahap analisis ini menentukan komponen dan integrasinya pada halaman. Komponen yang dikembangkan adalah komponen tampilan agar dapat dengan mudah digunakan kembali. Kemudian halaman yang dikembangkan adalah tampilan *website* sesuai rancangan UI dan integrasinya dengan komponen-komponen. Pada proses ini juga menganalisis penerapan Prinsip SOLID agar menciptakan komponen tampilan yang mudah dipelihara. Contoh untuk kasus ini adalah komponen Card:

1. SRP: Komponen Card hanya memiliki satu tanggung jawab untuk menampilkan dirinya sendiri.
2. OCP: Komponen Card dapat ditambahkan elemen tampilan lain seperti komponen Image dan Link.
3. LSP: Komponen yang ditambahkan pada komponen Card bersifat opsional, dapat ditambahkan jika perlu.
4. ISP: Komponen Card hanya menangani tampilan komponen Card, tidak dengan komponen yang ada di dalamnya.
5. DIP: Komponen Card diberi abtraksi, jadi menghindari ketergantungan implementasi dan memungkinkan nilai *default*.

TABEL 2 adalah daftar komponen-komponen yang dikembangkan lengkap dengan deskripsi dan Prinsip SOLID yang diterapkan. Kemudian, berdasarkan rancangan UI dari UI/UX *designer*, TABEL 3 adalah daftar halaman yang dikembangkan untuk tampilan fitur FeelsQuest mengacu pada FR ID dan terdapat komponen yang diintegrasikan.

1. Komponen

TABEL 2
(Komponen)

No	Komponen	Deskripsi	Prinsip SOLID
1	Card	Menampilkan konten dalam bentuk kartu yang terdiri dari beberapa bagian opsional seperti thumbnail, header, body, dan footer.	SRP, OCP, LSP, ISP, DIP
2	Button	Menampilkan tombol untuk menginisiasi aksi.	SRP, OCP, LSP, ISP, DIP
3	Link	Menampilkan tombol berisi tautan yang mengarahkan ke halaman lain.	SRP, OCP, LSP, ISP, DIP
4	Input	Menampilkan label sebagai informasi tentang elemen input.	SRP, OCP, LSP, ISP, DIP
		Menampilkan elemen input untuk menerima masukan.	
5	Image	Menampilkan gambar.	SRP, OCP, LSP, ISP, DIP
6	TopBar	Menampilkan ikon panah untuk navigasi kembali ke halaman sebelumnya.	SRP, OCP, LSP, ISP, DIP
		Menampilkan logo untuk identifikasi fitur.	
		Menampilkan judul halaman untuk memberi konteks tentang halaman yang sedang dilihat.	
		Menyediakan judul kursus(item) pada halaman.	

7	TextArea	Menampilkan label sebagai informasi tentang elemen input textarea. Menampilkan elemen textarea untuk menerima masukan.	SRP, OCP, LSP, ISP, DIP
8	Pagination	Menampilkan navigasi nomor halaman untuk membagi konten *website* menjadi beberapa halaman terpisah	SRP, OCP, LSP, ISP, DIP

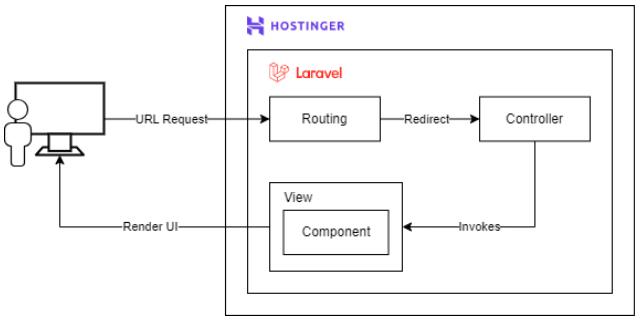
2. Halaman

TABEL 3
(Halaman)

No	FR ID	Halaman	Role	Komponen
1	FR01	Daftar Kursus	Guest	Card, Image, Link, Pagination
2	FR02	Detail Kursus		Card, Image, Link, TopBar
3	FR05	Akses Kursus dan Whiteborad	User	Card, Link, TopBar
4	FR06A, FR06B	Mengisi Feedback, Rating dan Ulasan Kursus		Card, Button, Link, TopBar
5	FR06C	Penyelesaian Kursus		TopBar, Link
6	FR01	Dashboard Daftar Kursus		Card, Image, Link, Pagination
7	FR02	Dashboard Detail Kursus		Card, Image, Link, TopBar
8	FR01, FR08	Dashboard Daftar Kursus	Admin	Card, Image, Link, Button, Input, TextArea, Pagination
9	FR09, FR10	Dashboard Detail Kursus		Card, Image, Link, Button, Input, TextArea
10	FR11, FR12, FR13, FR14	Dashboard Konten Kursus		Card, Button, TopBar
11	FR18	Dashboard Feedback Kursus		Card, Link, TopBar
12	FR15, FR16, FR17	Dashboard Pertanyaan Feedback Kursus		Card, Button, TopBar
13	FR18	Dashboard Daftar Feedback Kursus dari User		Card, Button, TopBar
14	FR09B, FR09C	Dashboard Rating dan Ulasan Kursus		Card, TopBar
15	FR09A	Dashboard Pendaftar Kursus		Card, TopBar

C. Rancangan

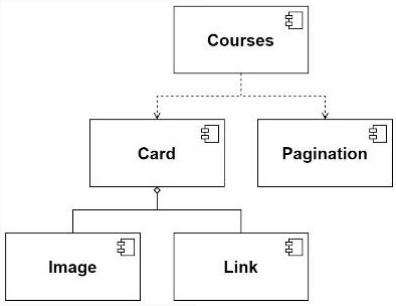
1. Desain Arsitektur



GAMBAR 4
(Rancangan Sistem)

GAMBAR 4 menggambarkan rancangan sistem dari pengembangan frontend FeelsQuest menggunakan *framework* Laravel. Pengguna mengakses URL pada *browser* yang dimana URL tersebut diterima oleh Routing pada aplikasi. Aplikasi meneruskan ke Controller yang ditentukan. Pada Controller dilakukan pengkodean logika yang dibutuhkan dan memanggil Views. Views tersebutlah yang ditampilkan di *browser website* pengguna. Pada Views terdapat Components yang dipanggil sebagai penyusun tampilan halaman. Pada Laravel rancangan ini terdapat pada app/View/Components yang dimana adalah rancangan yang otomatis dibuat oleh *framework* Laravel ketika membuat komponen baru.

2. Hierarki Komponen



GAMBAR 5
(Rancangan Hierarki Komponen pada Halaman Courses)

GAMBAR 5 menggambarkan rancangan hierarki komponen pada halaman Courses(Daftar Kursus). Rancangan hierarki ini menjelaskan ketergantungan dan keterkaitan komponen-komponen. Suatu komponen dapat bergantung dan berkaitan dengan satu atau lebih komponen untuk dapat menampilkan tampilan sesuai keutuhan.

Pada penelitian ini dicontohkan pada pengimplementasian komponen Card karena komponen ini sudah dapat mewakili pengimplementasian komponen lain.

D. Implementasi

Pada tahap ini komponen antarmuka dibuat menggunakan *framework* Laravel yang berfokus pada View. Fitur Laravel yang digunakan untuk pengimplementasian metode CBD adalah Laravel Blade Templates. Fitur ini dapat membuat komponen tampilan kustom dinamis yang dapat digunakan kembali. Dengan templating engine Blade yang dipakai pada View, tidak hanya kode HTML yang ditambahkan, namun juga memungkinkan untuk menampilkan *value* dari *variable*, membuat statements if else, melakukan iterasi data dan banyak lagi. Ini yang menjadikan Laravel lebih unggul dari

framework lain seperti React atau Vue, karena Blade memberikan kemudahan dalam integrasi langsung dengan logika back-end, memungkinkan penggunaan kode PHP dan struktur kontrol secara langsung dalam templat tanpa memerlukan proses build terpisah, sehingga mempercepat pengembangan dan mempermudah pemeliharaan aplikasi berbasis PHP. Dengan keunggulan tersebut Laravel layak digunakan pada penelitian ini.

Pada penelitian ini akan mencontohkan pengimplementasian dari Hierarki Komponen yang dibuat yaitu komponen Card sebagai komponen yang menampilkan kartu informasi ringkas kursus dan diintegrasikan pada halaman Courses.

1. Mengembangkan komponen

Laravel sangat membantu pengembang dalam mengembangkan komponen. Pada command prompt di proyek Laravel dapat dijalankan php artisan make:component NamaComponent untuk membuat *file* komponen yang terdapat di app/View/Components sebagai inisialisasi *class* dan juga pada resources/views/components sebagai pengkodean elemen HTML untuk tampilan. Pada tahap ini dilakukan pengkodean komponen pada TABEL 2 dengan prinsip SOLID yang ditentukan. Untuk membuat Komponen Card perlu menjalankan perintah php artisan make:component Components/Card. Kemudian pengkodean HTML dapat dilakukan di *file* resources/views/components/components/card.blade.php.

2. Menyusun komponen menjadi halaman

Pada tahap ini dibuat keseluruhan halaman aplikasi dengan mengintegrasikan komponen UI yang dibuat menggunakan Laravel sesuai hierarki komponen. Halaman dibuat dengan menjalankan perintah php artisan make:component Pages/Courses pada command prompt di proyek Laravel. Pada *file* resources/views/components/pages/courses.blade.php dapat dilakukan pengkodean untuk menyusun komponen yang telah dibuat. Komponen teresebut dipanggil di *file* halaman menggunakan *tag* permisalan. Contohnya komponen Card dipanggil di halaman Courses maka dapat dilakukan pemanggilan komponen dengan *tag* <x-components.card/>. Dengan begitu komponen Card dapat tampil di halaman Courses.

E. Evaluasi

Evaluasi dilakukan dengan pengujian *Maintainability* menggunakan *tools* PHPMetrics untuk mendapatkan nilai *Maintainability Index* (MI). MI digunakan untuk mengukur apakah *maintainability* dari pengembangan yang dilakukan bagus atau tidak. Jika nilainya tinggi maka pengembangan dengan metode CBD menggunakan Laravel ini dapat menghasilkan *maintainability* yang bagus. Dengan begitu dapat mempermudah perubahan, perbaikan dan penambahan tampilan pada pengembagan selanjutnya

IV. HASIL DAN PEMBAHASAN

A. Hasil Implementasi CBD

```php
<?php

namespace App\View\Components\Components;

use Closure;
use Illuminate\Contracts\View\View;
use Illuminate\View\Component;

class Card extends Component
{
    public $theme;
    public $image;
    public $link;

    /**
     * Create a new component instance.
     */
    public function __construct($theme = null, $image = null,  $link = null)
    {
        $this->theme = $theme;
        $this->image = $image;
        $this->link = $link;
    }

    /**
     * Get the view / contents that represent the component.
     */
    public function render(): View|Closure|string
    {
        return view('components.components.card');
    }
}
```

KODE PROGRAM 1
(Class Komponen Card)

```blade
@props([
    'image' => '',
    'header' => '',
    'body' => '',
    'footer' => '',
    'theme' => 'primary'
])

@php
  switch ($theme) {
    case 'secondary':
        $bordercolor = 'border-primary';
        $borderweight = 'border-2';
        $background = 'bg-body-tertiary';
        break;
    case 'tertiary':
        $bordercolor = 'border-primary';
        $borderweight = 'border-2';
        $background = 'bg-white';
        break;
    case 'primary':
    default:
        $bordercolor = '';
        $borderweight = 'border-0';
        $background = 'bg-white';
      break;
  }
@endphp

<div id="card" {{$attributes->merge(['style'=>'', 'class'=>''])}}>
    <div {{$attributes->merge(['style'=>'','class' => 'card shadow rounded w-100 ' . $bordercolor . ' ' . $borderweight . ' ' . $background])}}>
        {{ $image }}
        <div class="card-body mx-3">
            @if(!empty($header))
                <div id="header" {{ $header->attributes->merge(['style'=>'','class'=>'row'])}}>
                    {{ $header }}
                </div>
            @endif
            @if(!empty($body))
                <div id="body" {{ $body->attributes->merge(['style'=>'','class'=>'row'])}}>
                    {{ $body }}
                </div>
            @endif
            @if(!empty($footer))
                <div id="footer" {{ $footer->attributes->merge(['style'=>'','class'=>'row'])}}>
                    {{ $footer }}
                </div>
            @endif
        </div>
        {{$slot}}
    </div>
</div>
```

KODE PROGRAM 2
(Komponen Card)

```blade
<x-components.card class="col-12 col-lg-4 col-md-6 col-sm-12 text-center" style="max-height: 60vh">
    <x-slot:image>
        <x-components.image theme="thumbnail" :image="$course['course_photo_url']" :alt="$course['name']"/>
    </x-slot:image>
    <x-slot:header style="height: 8vh">
        <div class="col border-end border-primary border-2 my-auto">
            <p class="lead fw-bold text-primary m-0">@trimWord3($course['name'])</p>
        </div>
        <div class="col my-auto">
            <p class="lead text-primary m-0">@rupiah($course['price'])</p>
        </div>
    </x-slot:header>
    <x-slot:body style="height: 12vh">
        <h5 class="fs-6 text-primary my-auto">@trimWord10($course['brief_description'])</h5>
    </x-slot:body>
    <x-slot:footer>
        <x-components.link theme="primary" link="/feelsquest/{{ $course['id'] }}" text="Ikuti Kursus"/>
    </x-slot:footer>
</x-components.card>
```

KODE PROGRAM 3
(Penggunaan Komponen Card)

Hasil dari pengimplementasian CBD menggunakan Laravel ini adalah komponen-komponen tampilan contohnya yaitu KODE PROGRAM 2 Komponen Card. Komponen-komponen disusun pada halaman sesuai kebutuhan desain yang ada seperti KODE PROGRAM 3 Penggunaan Komponen Card.

Pada pengembangan ini, semua komponen dilakukan penerapan prinsip SOLID yang dimana penerapan ini sangat penting untuk membuat komponen memiliki pengkodean yang mudah dipelihara. KODE PROGRAM 1, KODE PROGRAM 2 dan KODE PROGRAM 3 adalah contoh bagaimana prinsip SOLID diterapkan pada kode komponen Card.

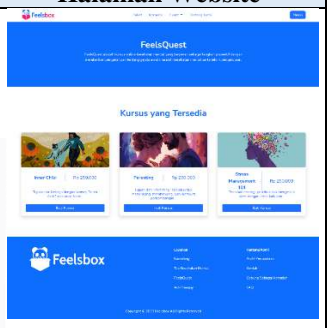
1. *Single Responsibility Principle*(SRP): KODE PROGRAM 1 menjelaskan Class komponen Card hanya memiliki satu metode render. Artinya Class ini hanya memiliki satu tanggung jawab yaitu menampilkan komponen Card. Komponen ini sesuai yang dideskripsikan pada TABEL 2 yaitu untuk menampilkan card yang berisi thumbnail, header, body, dan footer.
2. Open/Closed Principle(OCP): Komponen terbuka untuk perluasan(extension) tetapi tertutup untuk modifikasi. Hal ini dapat dicapai dengan menggunakan slot({{$slot}}) pada baris 50 di KODE PROGRAM 2, yang memungkinkan pengembang untuk menambahkan elemen tanpa harus memodifikasi komponen dasar secara langsung. Penambahan elemen dapat dilakukan pada baris setelah 18 di KODE PROGRAM 3 sebelum penutup komponen card.
3. *Liskov Substitution Principle*(LSP): Pada KODE PROGRAM 3, semua elemen pada slot image, header, body, dan footer dapat diubah sesuai kebutuhan dan bersifat opsional untuk digunakan. Perubahan atau tidak ditambahkannya elemen tidak akan memperburuk tampilan komponen Card.
4. *Interface Segregation Principle*(ISP): Pada KODE PROGRAM 1 Class Card memiliki extends Component yang dimana terdapat metode render yang bertindak seperti kontrak interface, jadi implement interface tidak dibutuhkan lagi. Selain itu di KODE PROGRAM 3 komponen Card memanggil komponen lain yaitu komponen Image pada KODE PROGRAM 3 baris 3 dan komponen Link pada KODE PROGRAM 3 baris 17. Ini bertujuan agar komponen Card tidak menjadi kompleks dan spesifik untuk melayani kebutuhan tampilan yang berbeda.
5. *Dependency Inversion Principle*(DIP): Pada KODE PROGRAM 1 baris 18 terdapat metode __construct untuk menginisialisasi nilai dari *variable* yang dibutuhkan dengan kata lain ini dapat bertindak sebagai abstraksi. Selain itu pada KODE PROGRAM 2 baris 1 komponen Card juga menggunakan @props untuk membuat abstraksi daripada detail implementasi. Ini membuat komponen Card tidak memiliki ketergantungan dan hanya bergantung pada abstraksi. @props juga memungkinkan komponen untuk memiliki nilai *default* jika tidak terdapat inputan dan dapat diubah secara dinamis berdasarkan kebutuhan.

GAMBAR 6 menampilkan hasil dari pengkodean komponen Card, yang dimana pada komponen tersebut terdapat image(Komponen Image), header, body, dan footer(Komponen Link).

GAMBAR 6
(Tampilan Komponen Card)

TABEL 4 memaparkan rancangan UI dengan hasilnya pada halaman. GAMBAR 7 adalah rancangan UI halaman Courses (Daftar Kursus) yang dibuat oleh UI/UX *designer*. Disebelahnya, GAMBAR 8 adalah hasil pengkodean halaman Courses dengan mengintegrasikan komponen Card. Ini menjelaskan bahwa metode CBD dapat menghasilkan halaman Courses sesuai dengan rancangan UI.

TABEL 4
(Rancangan UI Dengan Halaman Website)

Rancangan UI	Halaman Website
	
GAMBAR 7 (Rancangan UI Halaman Courses)	GAMBAR 8 (Hasil Halaman Courses)

B. Evaluasi Implementasi CBD

TABEL 5
(Nilai Pengujian *Maintainability*)

No	Komponen	Nilai MI
1	Button	97,42
2	Card	108,41
3	Image	97,42
4	Input	91,47
5	Link	101,15
6	Pagination	101,15
7	TextArea	97,42
8	TopBar	97,42
9	Admin\DashboardCourse	116,17
10	Admin\DashboardCourseContents	116,17
11	Admin\DashboardCourseFeedbackQuestions	116,17
12	Admin\DashboardCourseFeedbackUser	116,17
13	Admin\DashboardCourseFeedbackUsers	116,17

14	Admin\DashboardCourseParticipants	116,17
15	Admin\DashboardCourseReview	116,17
16	Admin\DashboardCourses	116,17
17	User\Course	116,17
18	User\CourseContents	116,17
19	User\CourseFeedback	116,17
20	User\Courses	116,17
21	User\CourseSertifikat	116,17
22	User\DashboardCourse	116,17
23	User\DashboardCourses	116,17
Rata-rata		110.20

TABEL 5 menampilkan hasil dari pengujian yang dilakukan secara otomatis menggunakan *tools* PHPMetrics untuk mendapatkan nilai *Maintainability Index* dari pengembangan antarmuka menggunakan Laravel. Semua komponen mendapatkan nilai lebih 85 dan dengan rata-rata 110.20, artinya pengembangan komponen tampilan ini memiliki *maintainability* yang bagus. Dengan begitu ini akan membuat kode mudah dipelihara sehingga memudahkan dalam perbaikan, perubahan atau penambahan fungsionalitas baru. Nilai tersebut didapatkan dengan penerapan metode CBD yang membuat komponen-komponen modular dan prinsip SOLID pada pengembangan tiap komponennya, yang dijelaskan pada poin 4.1.

V. KESIMPULAN

Pengembangan antarmuka banyak terjadi penulisan ulang kode program yang membuat pemeliharaan jadi sulit dilakukan padahal tampilan dan fungsi komponen yang dibuat relatif sama. Pada penelitian ini penulis menerapkan metode *Component-Based Development* (CBD) menggunakan Laravel dalam pengembangan antarmuka *website* fitur FeelsQuest agar dapat dengan mudah dilakukan pemeliharaan.

Penulis membuat komponen tampilan yang bertujuan agar dapat dipakai di banyak halaman. Terdapat 8 komponen yang dikembangkan dengan mengikuti prinsip SOLID untuk menciptakan kode yang mudah dipelihara. Komponen Card adalah salah satu contohnya. Kode program komponen Card pada satu *file* dapat dengan mudah digunakan pada halaman Courses atau halaman lain sesuai kebutuhan tampilan. Ini membuat komponen menjadi fleksibel, mudah dipelihara dan mengurangi penulisan ulang kode program.

Komponen yang telah dibuat diuji secara otomatis menggunakan *tools* PHPMetrics. Hasil pengujian berupa *Maintainability Index* (MI) yang digunakan sebagai pengukuran yang menyatakan bahwa kode program memiliki *Maintainability* yang baik artinya kemudahan dalam pemeliharaan jika terdapat perbaikan, perubahan dan penambahan tampilan baru.

Berdasarkan hasil dari penelitian ini, Nilai MI yang didapat memiliki nilai rata-rata lebih dari 85 yaitu 110.20 yang artinya pengembangan frontend fitur FeelsQuest dengan metode CBD dan penerapan prinsip SOLID menggunakan Laravel dapat menghasilkan *Maintainability* yang bagus. Hal ini membuktikan kode program dari komponen-komponen ini memiliki kemudahan dalam pemeliharaan. Sehingga pengembangan frontend dapat memudahkan untuk perbaikan, perubahan dan penambahan tampilan baru pada pengembangan selanjutnya.

REFERENSI

[1] M. Millenia, “Minimnya Kesadaran Masyarakat terhadap Mental Health,” Kementerian Kesehatan Direktorat Jenderal Pelayanan Kesehatan, 03 Agustus 2022. [Online]. Available: https://yankes.kemkes.go.id/view_artikel/974/minimnya-kesadaran-masyarakat-terhadap-mental-health. [Diakses 27 November 2023].

[2] M. A. Al-asaad, N. Selviandro dan E. Darwiyanto, “The Adoption of Component-Based Architecture in the Development of E-Learning Website Interface,” dalam The 1st International Conference on Software Engineering and Information Technology (ICoSEIT), 2022.

[3] S. U. Khan, A. W. Khan, F. Khan, M. A. Khan dan T. K. Whangbo, “Critical Success Factors of Component-Based Software Outsourcing Development From Vendors’ Perspective: A Systematic Literature Review,” IEEE Access, vol. 10, pp. 1650-1658, 2022.

[4] S. Aji, D. Pratmanto, A. Ardiansyah dan S. , “Implementasi Framework Laravel Dalam Perancangan Sistem Informasi Desa,” Indonesian Journal on Software Engineering (IJSE) Vol. 7, No. 2, 2021.

[5] I. M. A. Gunawan, K. E. Winarno, R. S. Y. Zebua dan Terttiaavini, “Perancangan dan Implementasi Frontend Web untuk Sistem Pengaduan Masyarakat,” Jurnal Informasi dan Teknologi Vol. 5 No. 1, 2023.

[6] Y. K, “Laravel Framework: Pengertian, Keunggulan & Tips untuk Pemula,” Niagahoster, 28 Juni 2019. [Online]. Available: https://www.niagahoster.co.id/blog/laravel-adalah/#Apa_itu_Laravel. [Diakses 28 November 2023].

[7] Eric, “Laravel: Komponen Blade 101,” dev.to, 20 Maret 2021. [Online]. Available: https://dev.to/ericchapman/laravel-blade-components-5c9c. [Diakses 30 Agustus 2024].

[8] N. Etrariadi dan E. S. P. A'inunisya, “Pengembangan Website Manajemen Proyek Menggunakan Metode Agile Scrum (Studi Kasus Diskopindag Kota Malang),” Jurnal Nasional Teknologi Dan Sistem Informasi - Vol. 09 No. 01, 2023.

[9] F. A, “Apa Itu Bootstrap? Pengertian, Fungsi, dan Kelebihannya,” Hostinger, 21 Februari 2023. [Online]. Available: https://www.hostinger.co.id/tutorial/apa-itu-bootstrap#Apa_Kelebihan_Bootstrap. [Diakses 27 November 2023].

[10] S. Ali, M. Abdellatief, M. A. Elfaki dan A. Wahaballa, “Complexity Metrics for Component-based Software Systems: Developer Perspective,” Indian Journal of Science and Technology, vol. 11(32), 2018.

[11] N. Padhy, R. Panigrahi dan S. C. Satapathy, “Identifying the Reusable Components from Component-Based System: Proposed Metrics and Model,” Information

Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing, vol 863, 2019.

[12] S. Kosasi dan David, "Penerapan Component Based Software Engineering Dalam Peangembangan Website Sistem Informasi Sekolah (Studi Kasus : Sma Santo Paulus Pontianak)," Seminar Nasional Informatika, pp. 507-512, 2012.

[13] V. K. Madasu, T. V. S. N. Venna dan T. Eltaeib, "SOLID Principles in Software Architecture and Introduction to RESM Concept in OOP," Journal of Multidisciplinary Engineering Science and Technology (JMEST), 2015.

[14] Mari dan Eila, "The impact of maintainability on component-based software systems," dalam 2003 Proceedings 29th Euromicro Conference, 2003.

[15] R. G. Atmaja, B. Priyambadha dan F. Pradana, "Pembangunan Kakas Bantu Untuk Mengukur Maintainability Index Pada Perangkat Lunak Berdasarkan Nilai Halstead Metrics dan McCabe's Cyclomatic Complexity," dalam Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 2019.