Implementasi Clean Code Pada Pengembangan Aplikasi Berbasis Web

by Jhon Doe

Submission date: 09-Jun-2021 06:34PM (UTC+0700)
Submission ID: 1603406916
File name: 122_MAKALAH.pdf (404.75K)
Word count: 1745
Character count: 11658

Implementasi *Clean Code* Pada Pengembangan Aplikasi Berbasis Web

Abstract*—Kerjasama tim adalah sebuah kemampuan yang harus dimiliki oleh suatu kelompok dalam mencapai tujuan yang sama. *Programmer* merupakan sebuah profesi yang bertanggung jawab dalam mengembangkan suatu sistem aplikasi. *Programmer* menuliskan kode program (Syntax) dan merancang sistem untuk mengembangkan suatu aplikasi. Seorang *programmer* harus memiliki kemampuan untuk berkolaborasi dengan tim untuk membuat suatu sistem. Untuk meningkatkan kemampuan *programmer* dalam bekerjasama dalam tim, *programmer* harus memahami kode yang dituliskan oleh *programmer* yang lain. Dengan konsep *Clean Code*, *programmer* dapat menuliskan sebuah kode yang mudah dibaca dan dimengerti untuk berkolaborasi dengan tim. *Programmer* perlu memperhatikan kode yang ditulis, menulis kode yang buruk akan berdampak ketika terjadi perubahan dan maintenance. Dengan menerapkan konsep clean code, kode akan menjadi lebih rapi, mudah dipahami dan terstruktur. Pada makalah ini akan dibahas tentang implementasi *clean code* pada pengembangan aplikasi berbasis web. Hasil implementasi *clean code* dapat membantu meningkatkan efektifitas serta efisiensi dalam melakukan pengembangan aplikasi serta memudahkan pengembang ketika terjadi perubahan terhadap kode.

Keywords—clean code, programmer, sistem, kode

I. PENDAHULUAN

Programmer merupakan sebuah profesi yang bertanggung jawab dalam mengembangkan suatu sistem aplikasi. *Programmer* menuliskan kode program (*Syntax*) dan merancang sistem untuk mengembangkan suatu aplikasi [1].

Programmer mengikuti konsep teori seperti algoritma, struktur data dan aspek paradigma pemrograman yaitu orientasi objek atau pemrograman fungsional. Dalam menulis kode, terdapat sebuah konsep bernama *clean code*, dengan mengikuti konsep tersebut *programmer* dapat menuliskan kode yang bersih dan mudah dipahami oleh *programmer* lain. Sebuah studi di antara 227 profesional IT menyatakan bahwa keterampilan yang harus dimiliki seorang programmer yaitu kemampuan untuk membaca, memahami, dan memodifikasi kode program yang ditulis oleh orang lain. Kualitas kode yang rendah memberikan pengaruh langsung terhadap perawatan sebuah program [2].

Kode yang buruk sangat mempengaruhi kualitas kode sebuah program. Kode yang buruk akan sangat susah untuk dikembangkan karena kode yang dibuat tidak beraturan dan tidak terstruktur sehingga sulit untuk dibaca dan dipahami. Kode yang sulit dibaca dan dipahami akan menghabiskan waktu untuk memahami maksud dan tujuan dari kode tersebut [3].

Menerapkan konsep Clean Code sangat berpengaruh dalam mengembangkan aplikasi, untuk berkolaborasi dalam sebuah *project* yang akan dikerjakan tim setiap *programmer* harus mengikuti konsep *clean code*, sehingga dengan kode yang baik dan benar antar *programmer* tidak mengalami kesulitan ketika melanjutkan kode dari *programmer* lain. Selain itu menerapkan *clean code* dapat menghemat waktu dan biaya saat harus dilakukan maintenance terhadap sistem [4].

Penelitian ini dimaksudkan untuk mengetahui cara implementasi clean code pada pengembangan aplikasi berbasis web PHP menggunakan *framework* Laravel.

II. KAJIAN PUSTAKA

A. Kode Program

Kode program yaitu suatu rangkaian pernyataan atau deklarasi yang ditulis dalam bahasa pemrograman komputer yang dapat dibaca oleh manusia. Kode program merupakan rangkaian kode yang tersusun untuk menjalankan instruksi yang dapat dipahami oleh komputer [5].

B. *Clean Code*

Menurut Robert, *clean code* adalah kumpulan kode yang mudah dibaca, mudah dimengerti dan mudah di *maintenance* oleh pengembangnya sendiri maupun orang lain [6]. *Clean code* diperlukan karena beberapa alasan, yang pertama adalah komunikasi, dengan membuat kode yang rapi dan benar programmer dapat menyampaikan dengan tepat maksud dari kode atau fungsi yang dibuat [7].

Menerapkan clean code dimulai dengan aturan penaaman, menghindari ambiguitas pada sebuah kondisi, menggunakan *coding style,* menghapus kode yang tidak digunakan dan dekomposisi program [8].

C. *Meaningful Names*

Penamaan yang baik sangat penting dalam menuliskan kode, baik untuk fungsi, kelas dan *variable*. Penamaan *variable* dalam komputer harus bermakna, ini dapat membantu pemahaman penulis kode program. Bermakna bisa diartikan sebagai nama yang deskriptif yang menjelaskan arti dari nama tersebut [9].

Nama yang bermakna dapat memudahkan pengembang lain untuk memahami maksud dan kegunaan dari nama tersebut. Contohnya seperti memberikan nama pada *variable* untuk sebuah tanggal, int d akan lebih sulit dipahami ketika dipanggil di kode lain, akan lebih mudah jika nama tersebut memiliki arti seperti int tanggal [10].

D. Fungsi

Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya. Dengan membuat fungsi, program menjadi terstruktur, sehingga mudah dipahami dan dikembangkan [11].

Dekomposisi kode bisa dilakukan dengan membuat sebuah fungsi, kode program yang panjang dipisahkan dengan sebuah fungsi berdasarkan jenisnya. Sehingga kode menjadi lebih sederhana dan mudah dipahami [12].

E. SonarLint

SonarLint adalah ekstensi IDE yang membantu mendeteksi dan memperbaiki masalah kualitas saat menulis kode program. Seperti pemeriksa ejaan, SonarLint menemukan kekurangan yang dapat diperbaiki pada sebuah kode.

SonarLint dapat membantu meningkatkan kualitas dari sebuah kode. SonarLint dapat membantu dalam menemukan *bugs* dan *issue*. SonarLint juga memberikan feedback tentang kode yang dituliskan [13].

F. PHP CS Fixer

PHP CS Fixer adalah *tools* untuk memperbaiki sebuah kode dengan mengikuti standar pengkodean bahasa pemrograman PHP yang mengikuti standar yang didefinisikan PSR-1, PSR-2 atau bisa mengikuti standar tim dengan melakukan konfigurasi manual [14].

Dengan PHP CS Fixer dapat membuat kode lebih mudah dibaca, hal-hal yang menentukan kode lebih mudah dibaca dengan memperhatikan, *indentation, line break* dan *whitespace* agar kode terlihat rapi. PHP CS Fixer dapat memperbaiki hal tersebut [15].

III. METODOLOGI

Penelitian dimulai dengan mempelajari konsep *clean code* dan penerapannya terhadap proyek. Tahap selanjutnya yaitu mengikuti *guidelines* pada perusahaan. *Guidelines* berisi aturan dasar dan ketentuan dalam penulisan *variable* dan menulis kode yang baik. Setelah memahami dan mengikuti *guidelines* tahap selanjutnya yaitu mengimplementasikan aturan dasar yang telah ditentukan berdasarkan pemahaman dan berkonsultasi dengan anggota tim proyek.

IV. HASIL DAN PEMBAHASAN

A. Penamaan

Nama sangat penting dalam menuliskan sebuah kode, nama *variable*, fungsi atau kelas yang bermakna dapat menjelaskan apa yang dimaksud dalam baris kode. Nama dengan konteks dan makna yang jelas dapat memberikan arti pada keseluruhan baris kode. Tabel 1 berikut adalah penulisan nama yang ditetapkan oleh perusahaan [16].

TABLE I. ATURAN PENAMAAN

What	Penulisan	Contoh
Variable	*camelCase*	$userId
Class Property	*camelCase*	private $accessToken
Class Method	*camelCase*	$postRepository->featuredArticle()
Model	*StudlyCase*	User UserProfile

Dalam penulisan *variable*, fungsi dan *method* menggunakan penulisan *camelCase* sedangkan untuk model menggunakan penulisan *StudlyCase*. Dalam menuliskan nama kelas atau model, pastikan menggunakan kata benda seperti *Package*, Instansi. Untuk penamaan *method* menggunakan kata kerja seperti *getDataCustomer* dan *getDataDukung*.

Gambar 1 menunjukan penamaan *variable* dengan penulisan *camelCase* dengan 2 kata, kata pertama diawali dengan huruf kecil yang dilanjutkan dengan kata kedua dengan diawali huruf besar tanpa terpisah. Untuk penamaan *variable* dengan 1 kata menggunakan huruf kecil.

Gambar 2 menunjukan penamaan kelas model dengan penulisan *StudlyCase* dengan 2 kata, kedua kata menggunakan huruf besar tanpa dipisah.

Gambar 3 menunjukan penaaman kelas *method,* aturan penamaan kelas *method* sama dengan aturan penamaan pada *variable* dengan menggunakan penulisan *camelCase*. Dalam penulisan nama lebih detailnya ada di Gambar 1 , Gambar 2 dan Gambar 3.

```
public function create(Verification $verification)
{
    $komitmenInstansi = KomitmenInstansi::with( relations: 'bukti
                                                ->where( column: 'packa
                                                ->where( column: 'insta
    $history = Verification::with( relations: 'assignVerificator
                                                'rekomendasiSement
                                                'rekomendasiSement
```

Gambar 1. Nama *Variable*

```
class PackageInstansi extends Model implements HasMedia
{
    use AutoSearch, AutoSort, AutoFilter, softDeletes, InteractsWithMedia;

    protected $table = 'package_instansi';
    protected $fillable = ['package_id', 'instansi_code', 'status', 'file_komitmen'];

    protected $searchableColumns = ['instansi.name'];

    protected static function boot(){...}

    public function package(){...}

    public function instansi(){...}

    public function verifikator(){...}

    public function masterData(){...}
}
```

Gambar 2. Nama Fungsi dan Model

```
public function destroyDataDukung($komitmen)
{
    return DataDukung::where( column: 'komitmen_id', $komitmen->id)->delete();
}

public function getDataDukung($komitmen)
{
    return DataDukung::where( column: 'komitmen_id', $komitmen->id)->with( rela
}
```

Gambar 3. Nama *Method*

B. Duplikasi Kode

Suatu fungsi harus melakukan satu hal, jika suatu fungsi melakukan banyak hal, dapat menyebabkan kebingungan. Setiap pernyataan dalam fungsi tidak boleh lebih dari satu, karena fungsi memiliki tujuan yang jelas [17].

Apabila kode tidak terbungkus dalam suatu *method* atau fungsi, jika terjadi suatu perubahan maka semua kode harus diubah. Dengan adanya fungsi dan *method*, kode menjadi lebih terstruktur, dapat mengurangi kode yang terduplikasi. Kode program yang sama dan dipakai berulang dapat dituliskan sekali secara terpisah dalam bentuk fungsi dan *method*. *Refactor* kode menjadi lebih mudah.

Pada fungsi *create* dan edit terdapat sebuah kode untuk mengambil data, karena data yang diambil sama, maka dibuatlah sebuah *method getPackage* dan *getKomitmen*, dengan menggunakan *method* kode yang sama tidak perlu ditulis ulang di kedua fungsi tersebut, sehingga dapat mengurangi duplikasi kode.

Gambar 4 menunjukan fungsi *create* dan edit, di dalam fungsi tersebut terdapat sebuah kode untuk memanggil sebuah method yang sama.

```
public function create()
{
    $package = $this->service->getPackage();
    $komitmens = $this->service->getKomitmen();
    $dataPackageKomitmenId = [];

    return view( view: 'package-komitmen::create', comp
}

public function edit($packageId)
{
    $package = $this->service->getPackage();
    $komitmens = $this->service->getKomitmen();
    $dataPackageKomitmen = $this->service->getDataPack
    $dataPackageKomitmenId = $dataPackageKomitmen->kom
```

Gambar 4. Fungsi *Create* dan Edit

Gambar 5 menunjukan kode yang ada pada setaip method untuk mengambil sebuah data.

```
public function getPackage()
{
    $package = Package::all();

    return $package->pluck( value: 'nama_package', key: 'id');
}

public function getKomitmen()
{
    return Komitmen::all();
}
```

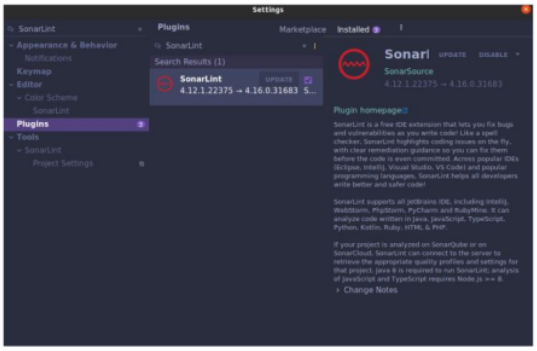
Gambar 5. *Method* getPackage dan getKomitmen

C. Kualitas Kode

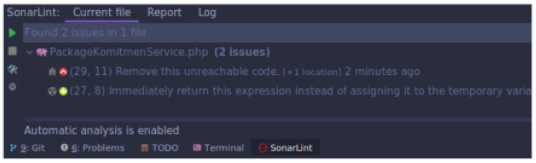
Kualitas kode mendefinisikan kualitas seorang *programmer* dalam menuliskan kode. Untuk meningkatkan kualitas kode terdapat *tools* SonarLint yang dipasang di *IDE* dan PHP CS Fixer yang dipasang pada sistem

Pada *tool* SonarLint dapat dipasang di *IDE* seperti pada Gambar 6 menunjukan bahwa SonarLint sudah terpasang pada *IDE*. Dengan menjalankan SonarLint untuk melakukan pengujian, Sonarlint akan menampilkan *error* pada kode jika terdapat sebuah kesalahan.

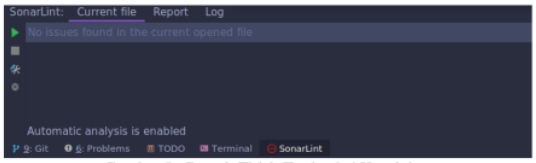
Gambar 7 menampilkan sebuah kesalahan yang terdeteksi dengan SonarLint. SonarLint akan menunjukan pesan kesalahan. Pada Gambar 7 dijelaskan bahwa ada sebuah kode yang tidak terpakai dan tidak dapat dijalankan, sehingga kode dapat dihapus atau diperbaiki. Pada Gambar 8 tidak menampilkan kesalahan kode yang terdeteksi pada SonarLint.



Gambar 6. SonarLint



Gambar 7. Contoh Kesalahan Terdeteksi



Gambar 8. Contoh Tidak Terdeteksi Kesalahan

V. KESIMPULAN

Dalam implementasi *Clean Code* pada pengembangan aplikasi berbasis web dapat disimpulkan sebagai berikut:

- Dengan adanya konsep *clean code programmer* dapat membaca dan memahami sebuah kode.
- Memudahkan pekerjaan *programmer lain*, ketika kode yang dituliskan akan dilanjutkan oleh *programmer* lain.
- Memudahkan pekerjaan, ketika terjadi perubahan pada sebuah kode, kode yang mudah dipahami akan sangat mudah untuk dilakukan perubahan.

- Menghemat waktu, dengan menggunakan *tools* dapat mempercepat waktu dalam menemukan kesalahan dalam sebuah kode karena dilakukan secara otomatis. Dengan kode yang tersusun dan terstruktur melakukan perubahan dapat dilakukan dengan cepat karena kode mudah dipahami.
- Dapat mengurangi duplikasi kode, ketika kode yang sama akan dijalankan berulang.
- Jika terjadi sebuah bug dapat ditemukan dengan mudah karena kode mudah dipahami.
- Meningkatkan kerjasama tim.

REFERENCES

[1] Supriyatna Adi, "TEKNIKA," *Sistem Informasi Forum Diskusi Programmer Berbasis Web Menggunakan Rapid Application Development*, vol. 7, no. 2, November 2018.

[2] W. Dietz Linus, "Teaching Clean Code," *Teaching Clean Code*.

[3] Sunny Sultan. (2019, October) Noteworthly-The Journal Blog. [Online]. https://blog.usejournal.com/what-is-bad-code-how-to-write-clean-code-a9b7b539ad8

[4] Wellcode.io Team. (2020, February) WSINSIGHT. [Online]. https://insight.wellcode.io/clean-code

[5] Vinashaw. (2018, February) dictio. [Online]. https://www.dictio.id/t/apa-yang-dimaksud-dengan-source-code-atau-kode-program/15085

[6] Muhammad Zaky. (2020, February) Meduim. [Online]. https://medium.com/itmi-engineering/apa-itu-clean-code-dan-kenapa-begitun-penting-part-1-3e0fce6984c

[7] LIMANTARA NATALIA. (2014, April) Binus Univeristy. [Online]. https://sis.binus.ac.id/2014/04/12/clean-code/

[8] Didik Tri Susanto. (2018, September) DOT Blog. [Online]. https://blog.dot.co.id/5-cara-mudah-menerapkan-clean-code-b2e0ec1b860e

[9] Gal.dkk Beniamini. (2017, June) https://ieeexplore.ieee.org/abstract/document/7961503. [Online]. https://ieeexplore.ieee.org/abstract/document/7961503

[10] Daniyal Ahmad Rizaldhi. (2019, October) elibrary UNIKOM. [Online]. https://elibrary.unikom.ac.id/id/eprint/1119/8/UNIKOM_Daniyal%20Ahmad%20Rizaldhi_BAB%202.pdf

[11] Yuliana. (2009, Agustus) Konsep Pemrograman. [Online]. http://yuliana.lecturer.pens.ac.id/Konsep%20Pemrograman/Teori/T7-Fungsi(1).pdf

[12] Andi Taru NNW. (2019, Desember) GAMELAB INDONESIA. [Online]. https://www.gamelab.id/news/153-perhatian-programmer-inilah-pentingnya-menerapkan-clean-code

[13] Dikih Arif Wibowo. (2020, November) Javan Cipta Solusi. [Online]. https://blog.javan.co.id/meningkatkan-code-quality-dengan-plugin-sonarlint-di-intellij-idea-36705b6cd8fa

[14] Dariusz Ruminski. (2021, June) Github. [Online]. https://github.com/FriendsOfPHP/PHP-CS-Fixer

[15] dinar. (2020, August) Tekno Sejahtera. [Online]. https://teknosejahtera.com/sebuah-seni-menerapkan-clean-code/

[16] Javan. laravolt. [Online]. https://laravolt.dev/docs/v4/guidelines/naming-things/

[17] Henning Grimeland Koller. (2016) University of Oslo. [Online]. https://www.duo.uio.no/bitstream/handle/10852/51127/master.pdf

Implementasi Clean Code Pada Pengembangan Aplikasi Berbasis Web

ORIGINALITY REPORT

10%	10%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	sis.binus.ac.id Internet Source	4%
2	ppratama499.wordpress.com Internet Source	2%
3	www.dictio.id Internet Source	1%
4	laravolt.dev Internet Source	1%
5	www.scribd.com Internet Source	1%
6	medium.com Internet Source	1%
7	ejournal.ikado.ac.id Internet Source	1%
8	www.slideshare.net Internet Source	1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		