https://journaledutech.com/index.php/great
Global Research and Innovation Edutech Journal (GREAT)
Vol. 01, No. 01, 2025, Hal. 346-354

ANALISIS DAN PENERAPAN CLEAN CODE PADA APLIKASI SIMUDAH BERDASARKAN CODE SMELLS DAN HEURISTIK

Marcelo Amazona[1], Wijang Widhiarso[2], Muhammad Rachmadi[3]
Sistem Informasi, Universitas Multi Data Palembang, Palembang
E-mail: *amazonamarcelo@mhs.mdp.ac.id[1], wijang@mdp.ac.id[2], rachmadi@mdp.ac.id[3]

ABSTRAK

Pengembangan perangkat lunak dalam era kemajuan teknologi menuntut penerapan prinsip Clean Code untuk memastikan keterbacaan dan efisiensi kode. Penelitian ini fokus pada aplikasi SIMUDAH dari Universitas Multi Data Palembang, dikembangkan dengan framework Laravel. Meskipun framework tersebut memberikan struktur yang terorganisir, SIMUDAH menghadapi kendala pemeliharaan, seperti singkatan variabel dan dead code. Penelitian ini bertujuan menerapkan Clean Code dengan mengidentifikasi dan memperbaiki smells code, menggunakan heuristics seperti Single Responsibility Principle dan Encapsulation. Diharapkan, penelitian ini akan meningkatkan kualitas dan pemeliharaan SIMUDAH, menjadikannya solusi yang relevan dan efisien di lingkungan kegiatan mahasiswa.

Kata kunci ***Clean Code, Code Smells*, Laravel, Pemeliharaan Kode, Efisiensi kode**

ABSTRACT

Software development in the era of technological advancement demands the implementation of Clean Code principles to ensure code readability and efficiency. This research focuses on the SIMUDAH application from the Universitas Multi Data Palembang, developed using the Laravel framework. Despite the organized structure provided by the framework, SIMUDAH faces maintenance challenges, such as the use of variable abbreviations and dead code. This research aims to implement Clean Code by identifying and rectifying code smells, applying heuristics such as the Single Responsibility Principle and Encapsulation. It is anticipated that this study will enhance the quality and maintenance of SIMUDAH, making it a relevant and efficient solution in the context of student activities.

Keywords ***Clean Code, Code Smell, Laravel, Code Maintenance, Code Efficiency***

1. PENDAHULUAN

Dalam era perkembangan teknologi yang semakin maju, pengembangan perangkat lunak menjadi kritis dan memerlukan fondasi kuat dalam penerapan prinsip-prinsip Clean Code (F.hadiansyah, 2019). Clean Code, sebagaimana dijelaskan dalam buku "Clean Code" oleh Robert C. Martin, menjadi kunci utama dalam membangun perangkat lunak inovatif dan berkualitas tinggi. Prinsip-prinsip ini penting untuk menjaga keterbacaan, pemeliharaan, dan efisiensi kode (I.Liem, 2010).

"Dampak Kode" atau indikasi bahwa kode mungkin tidak memenuhi standar Clean Code, menjadi isyarat awal dalam konteks zaman yang terus berkembang (S.Parera, 2016). Fenomena ini mencakup fungsi yang terlalu panjang, penggunaan variabel dengan nama yang tidak jelas, komentar yang kurang informatif, atau kompleksitas berlebih pada kelas. Dalam menghadapi kemajuan teknologi, penerapan prinsip-prinsip Clean Code menjadi fokus utama untuk menjaga relevansi dan kualitas perangkat lunak (I.Liem, 2010).

Tantangan muncul ketika prinsip-prinsip ini diterapkan dalam konteks pengembangan yang terus berkembang pesat. Heuristik atau aturan praktis, seperti Single Responsibility Principle dan Encapsulation, menjadi panduan penting dalam menghadapi kompleksitas

pengembangan perangkat lunak di era canggih ini (M.lidnillah, 2008). Heuristik ini membantu pengembang membuat keputusan lebih baik dalam menulis, membaca, dan memelihara kode, serta menjadi kunci sukses dalam menghadapi tuntutan pengembangan perangkat lunak di zaman modern ini.

Universitas Multi Data Palembang (UMDP) menjadi lembaga pendidikan progresif yang menyikapi tuntutan zaman. Sebagai yayasan pendidikan, UMDP fokus pada transfer pengetahuan, keterampilan, dan nilai dari pengajar kepada pelajar. Untuk memudahkan kegiatan mahasiswa, UMDP mengembangkan aplikasi bernama SIMUDAH (Sistem Informasi MDP Unit Kegiatan Mahasiswa dan Himpunan Mahasiswa Jurusan). SIMUDAH memiliki peran penting dalam mengelola kegiatan mahasiswa, termasuk absensi dan pelaporan.

Pentingnya aplikasi SIMUDAH semakin meningkat karena dikembangkan menggunakan framework Laravel, yang dikenal dengan kejelasan dan kelincahan kodenya. Dengan menggunakan Laravel, SIMUDAH tidak hanya efisien dalam mengelola kegiatan mahasiswa tetapi juga memastikan basis kode mengikuti prinsip-prinsip Clean Code. Penggunaan framework Laravel memberikan struktur terorganisir dan standar pengkodean yang jelas, memudahkan pemahaman, pengelolaan, dan pemeliharaan source code SIMUDAH.

Namun, SIMUDAH menghadapi beberapa permasalahan terutama dalam pemeliharaan. Kode yang penuh dengan singkatan variabel dan dead code menjadi hambatan utama dalam pemahaman dan pemeliharaan sistem ini. Tantangan ini menciptakan dampak negatif terhadap keterbacaan dan pemeliharaan SIMUDAH, mengingat semakin kompleksnya kebutuhan dan ekspektasi dalam pengembangan perangkat lunak di era modern.

Dengan menyadari tantangan ini, penulis mengusulkan penerapan *Clean Code* pada SIMUDAH, fokus pada identifikasi dan perbaikan *smells code*, serta penerapan heuristik untuk menciptakan kode yang bersih, mudah dipahami, dan mudah dikelola. Melalui pendekatan ini, diharapkan SIMUDAH tetap menjadi solusi yang relevan, efisien, dan andal dalam mengelola kegiatan mahasiswa di tengah kemajuan zaman. Dengan demikian, judul penelitian ini adalah "Analisis dan Penerapan *Clean Code* pada Aplikasi SIMUDAH Berdasarkan *Code Smells* dan *Heuristik*."

2. METODE PENELITIAN

Dalam Penelitian ini, digunakan jenis penelitian dengan pendekatan kualitatif, sejalan dengan metodologi yang sering diterapkan dalam penelitian berbasis jurnal. Penelitian ini melibatkan sejumlah tahap, dimana setiap tahap didukung oleh metode metode tertentu. Metodologi penelitian yang diadopsi dalam penelitian ini terinspirasi oleh buku berjudul "*Clean Code: A Handbook of Agile Software Craftsmanship*," yang merupakan karya dari *Robert C Martin,* dan penyesuain dilakukan sesuai kebutuhan penelitian.

Gambar 1 Metode Penelitian

2. 1 Studi Literatur

Peneliti memperoleh dasar teori dari sumber sumber seperti jurnal internasional, buku, artikel, dan penelitian sebelumnya pedoman dalam merancang penelitian ini.

2. 2 Wawancara dan kesepakatan *source code*

Pada *fase* ini, dilakukan interaksi dengan pembuat aplikasi untuk memberikan dukungan selama penelitian *Cleancode* pada aplikasi yang telah disetujui.

2. 3 Identifikasi dan seleksi Fitur

Pada langkah ini, kami melakukan seleksi fitur yang paling cocok dengan tujuan penelitian ini, dengan memperhatikan perbaikan yang signifikan dalam kode.

2. 4 Analisis *Code Smells*

Analisis menyeluruh dilakukan pada kode sumber untuk mengindentifikasi potensi *code smells* yang mungkin ada. Langkah ini krusial untuk menentukan wilayah yang memerlukan perbaikan.

2. 5 Implementasi *Refactoring* dengan prinsip *heuristik*

Refactoring diterapkan dengan mematuhi prinsip-prinsip *heuristik* yang telah ditetapkan. Pendekatan ini membantu memastikan bahwa perubahan yang dilakukan tidak hanya meningkatkan kualitas kode, tetapi juga menjaga integritasnya.

2. 6 Evaluasi hasil *Refactoring*

Pada tahap akhir, kami melakukan evaluasi komprehensif terhadap hasil *refactoring,* dengan mengukur peningkatan dalam kualitas kode.

3. HASIL DAN PEMBAHASAN

3. 1 Studi Literatur

a. *Code smells*

Code smell adalah istilah yang digunakan untuk menunjukkan adanya potensi masalah dalam sebuah kode program (A.J.Paramita, 2022). Ketika kode dikategorikan sebagai mengandung *code smell*, ini mengindisikasikan bahwa ada hal yang tidak beres atau seharusnya tidak ada dalam kode tersebut (I.Liem, 2010).Dalam bukunya yang sangat berguna, *Refactoring Clean code*, Robert C. Martin mengidentifikasi berbagai jenis *code smells* yang dapat ditemukan dalam kode *program*. Bob martin memberikan panduan untuk menganali dan mengatasi *code smells* ini, sehingga membantu meningkatkan kualitas kode. Maka dari itu, untuk memudahkan referensi dan penerapan heuristik ini, sebuah daftar panduan heuristik telah dibuat Robert C. Martin (R.C.Martin, 2016).

b. *Clean Code*

Menurut Robert C. Martin (sering disebut sebagai "Uncle Bob"), konsep "clean code" merujuk pada sekumpulan kode yang memiliki sifat-sifat berikut: mudah dibaca, mudah dimengerti, dan mudah untuk dikelola atau dipelihara oleh pengembang yang menulisnya maupun oleh pengembang lainnya (R.C.Martin, 2016). Clean code menjadi penting karena beberapa alasan. Pertama, aspek komunikasi; dengan menulis kode yang bersih dan benar, seorang programmer dapat dengan jelas menyampaikan maksud dari kode atau fungsi yang dibuat. Alasan kedua adalah kolaborasi; kode yang baik dan benar akan memudahkan programmer dalam berkolaborasi karena kode tersebut dapat dengan mudah dipahami dan dibaca oleh tim pengembangan (M.idris, 2021).

c. *Heuristics*

Menurut Schoenfeld (1980), "heuristik" merujuk pada saran atau strategi umum bagi pemecah masalah tanpa keterkaitan khusus dengan topik tertentu. Heuristik berfungsi sebagai panduan umum untuk mendekati masalah dan efisien mengalokasikan sumber daya. Berbeda dengan algoritma matematika yang terstruktur, heuristik adalah pendekatan umum dalam pemecahan masalah, memberikan "peta jalan" tanpa langkah-langkah berurutan. Meskipun solusi heuristik tidak selalu mutlak benar atau optimal, tujuannya adalah memberikan arah dalam memecahkan masalah (M.lidnillah, 2008).

d. *Model analisis Aplikasi*

Model analisis merupakan representasi variabel-variabel yang digunakan dalam melakukan analisis data dengan tujuan memperoleh kesimpulan (Gunawan, 2003).

e. *PHP*

PHP, singkatan dari PHP: Hypertext Preprocessor, merupakan bahasa pemrograman yang digunakan untuk mengembangkan website dinamis dan aplikasi web. Berbeda dengan HTML yang hanya menampilkan konten statis, PHP mampu berinteraksi dengan database, file, dan folder, memungkinkan tampilan konten yang dinamis. Contoh aplikasi web yang dapat dibuat dengan PHP meliputi blog, toko online, CMS, forum, dan jejaring sosial. PHP adalah bahasa scripting, bukan berbasis tag seperti HTML, serta bersifat cross-platform, dapat dijalankan pada berbagai sistem operasi seperti Windows, Linux, dan Mac. Program PHP ditulis dalam file teks dengan ekstensi ".php" (T.yulianto, 2017).

f. *Laravel*

Laravel merupakan framework aplikasi website dengan syntax yang ekspresif dan elegan. Kerangka kerja pada web menyediakan struktur dan titik awal membuat aplikasi dengan detail merupakan bawaan dari laravel. Fitur-fitur yang disediakan seperti dependency injection menyeluruh, layer abstraksi database ekspresif, antrian dan pekerjaan terjadwal, pengujian unit dan integrasi, dan banyak lagi (A.J.Paramita, 2022).

g. *Visual Studio Code*

Visual Studio merupakan kumpulan alat pengembangan dari Microsoft yang dirancang untuk membuat aplikasi enterprise dan kelengkapannya. Terdapat lima alat utama dalam Visual Studio, yaitu Visual Basic, Visual C++, Visual Interdev, Visual Foxpro, dan Visual J++ (Gunawan, 2003).

3. 2 Identifikasi Aplikasi Simudah dan Fitur

Aplikasi SIMUDAH, sebuah sistem manajemen kegiatan dan absensi, dikembangkan khusus untuk penggunaan di lingkungan universitas, terutama dalam kegiatan Unit Kegiatan Mahasiswa (UKM) dan Himpunan Mahasiswa Jurusan (HMJ). Aplikasi ini dirancang untuk mempermudah pengelolaan dan pemantauan kehadiran anggota organisasi mahasiswa serta pelatih dalam berbagai kegiatan. Mengintegrasikan fitur-fitur seperti Absensi, Absensi Detail, Anggota, Jadwal, Kegiatan, Ketua Mhs, Laporan, Login, Pelatih, Pembina, Pengumuman, dan Ukm, aplikasi ini memiliki total kode sebanyak 264.015 baris dan menggunakan framework Laravel. Melalui proses refactoring, fitur-fitur kunci seperti Absensi, Jadwal, Anggota, Laporan, Login, dan Laporan Mahasiswa telah diperbarui dan ditingkatkan, membuktikan bahwa SIMUDAH adalah proyek pengembangan perangkat lunak yang signifikan dan penting untuk pemantauan kegiatan organisasi mahasiswa, seperti yang terlihat pada gambar 2.

Date : 2023-10-06 23:18:41

Directory c:\Users\Lenovo\Documents\PTA & TA\simudah\simudah

Total : 356 files, 201776 codes, 27316 comments, 34923 blanks, all 264015 lines

Summary / Details / Diff Summary / Diff Details

Languages

language	files	code	comment	blank	total
JavaScript	112	80,882	25,103	19,016	125,001
CSS	41	61,160	235	14,052	75,447
SQL	1	34,985	230	119	35,334
XML	4	9,328	14	4	9,346
PHP	191	8,151	1,733	1,704	11,588
JSON	3	7,201	0	3	7,204
Markdown	1	44	0	22	66
YAML	1	13	0	1	14
HTML	1	12	0	1	13
SCSS	1	0	1	1	2

Gambar 2 Jumlah baris kode

3. 3 Analisis *Code Smells*

Dalam pengembangan perangkat lunak, code smell seringkali menjadi tantangan yang dihadapi oleh para pengembang. *Code smell* merupakan indikasi potensi masalah dalam desain atau *implementasi* suatu kode *program*. Contoh *code smell* meliputi *duplicated code*, *long method*, dan dead code, masing-masing dengan karakteristik dan ciri-ciri deteksi yang berbeda.

Analisis code smell pada suatu proyek melibatkan identifikasi jenis *code smell* yang mungkin muncul. Sebagai contoh, ditemukan enam kasus *code smell Chose Descriptive Names* (N1), di mana penamaan *variabel*, fungsi, atau elemen kode tidak deskriptif. Pemilihan nama yang lebih jelas dapat meningkatkan keterbacaan dan pemeliharaan kode. Selain itu, ditemukan 19 kasus *Commented-Out Code* (C5), yang mencerminkan kode yang tidak aktif dan dapat dioptimalkan untuk meningkatkan kejelasan kode.

Satu kasus *Dead Code* (G9) menunjukkan adanya kode yang tidak lagi digunakan. Membersihkan kode mati dapat mengurangi kompleksitas dan meningkatkan pemahaman. Analisis *code smell* seperti ini penting untuk merancang strategi perbaikan yang tepat, sehingga pengembangan perangkat lunak dapat dilakukan dengan lebih efisien dan efektif.

```
81      public function show($id)
82      {
83          $ukm = Ukm::find($id);
84          $anggota = Anggota::where(['ukm_id'=>$id])->where('status', '=', 'Aktif')->get();
85          return view('anggota.index', compact('anggota','ukm'));
86      }
87
88      public function showall($id)
89      {
90          $ukm = Ukm::find($id);
91          $anggota = Anggota::where(['ukm_id'=>$id])->get();
92          return view('anggota.showall', compact('anggota','ukm'));
93      }
```

Gambar 3 *Code* Teridentifikasi *Duplicate*

Gambar di atas menunjukkan contoh kasus duplikasi kode yang melanggar heuristik yang dijelaskan dalam buku "Clean Code" karya Robert C. Martin.

```php
public function store(Request $request)
{
    $request->validate([
        'ukm_id'        => ['required', 'unique:jadwal,ukm_id'],
        'waktu_mulai'   => ['required'],
        'waktu_selesai' => ['required'],
        'hari'          => ['required'],
        'tempat'        => ['required']
    ], [
        'required'      => 'Field harus diisi!',
        'unique'        => 'Jadwal sudah ada!'
    ]);

    $jadwal = new Jadwal;
    $jadwal->ukm_id = $request->ukm_id;
    $jadwal->waktu_mulai = $request->waktu_mulai;
    $jadwal->waktu_selesai = $request->waktu_selesai;
    $jadwal = $request->merge([
        'hari' => implode(', ', (array) $request->get('hari'))
    ]);
```

Gambar 4 *Code* Teridentifikasi *Function should do one thing*

Gambar diatas menunjukan contoh kasus *function* memiliki tugas lebih dari satu yang melanggar *heuristik* yang dijelaskan dalam buku *Clean Code* karya Robert C martin.

```php
public function show(Jadwal $jadwal)
{
    //
}
```

Gambar 5 Code Teridentifikasi *Dead Code*

Gambar diatas menunjukkan contoh kasus *Function* Tidak memiliki isi atau fungsi yang melanggar *heuristik* yang dijelaskan dalam buku *Clean Code* karya Robert C martin.

```php
// if(!empty($request->kehadiran_pelatih)) {
//     if (count(explode(',', $request->pelatih_id)) > 1) {
//         $pelatih = json_decode($request->pelatih_id);
//         foreach($pelatih as $idpelatih) {
//             $laporan = new Laporan;
//             $laporan->ukm_id = $request->ukm_id;
//             $laporan->pelatih_id = $idpelatih;
//             $laporan->kehadiran = $request->kehadiran_pelatih[$idpelatih];
//             $laporan->save();
//         }
//     } else {
//         $laporan = new Laporan;
//         $laporan->ukm_id = $request->ukm_id;
//         $laporan->pelatih_id = $request->pelatih_id;
//         $laporan->kehadiran = $request->kehadiran_pelatih;
//         $laporan->save();
//     }
// }
```

Gambar 6 *Code* Teridentifikasi *Commented Out Code*

Gambar diatas menunjukkan contoh kasus *Commented Out Code* yang dimana *code* yang tidak terpakai dijadikan *Comment.*

```php
@foreach ($results as $a)
```

Gambar 7 *Code* Teridentifikasi *Choose Name Descriptive*

Gambar diatas menunjukkan contoh kasus *Choose Name Descriptive* yang dimana *code* yang nama *variable* yang tidak deskriptif.

3. 4 *Refactoring*

Pada tahap ini, dilakukan refactoring terhadap kode yang teridentifikasi pada fitur-fitur tertentu. Berikut adalah rincian refactoring yang dilakukan:

Refactoring Pada Fitur Absensi

Controller Fitur Absensi:

Terdapat 3 kasus "Commented-Out Code (C5)" yang dihapuskan, dan 4 kasus "Dead Code (G9)" yang diatasi.

Resource Views Fitur Absensi:

Ditemukan 4 kasus "Commented-Out Code (C5)" dan 2 kasus "Chose Descriptive Names (N1)." Semua komentar yang tidak relevan dihapuskan, dan penamaan yang tidak deskriptif diperbaiki.

Refactoring pada Fitur Anggota:

Controller Fitur Anggota:

Terdapat 1 kasus "Commented-Out Code (C5)" yang dihapuskan, dan 1 kasus "Duplication (G5)" yang diatasi.

Resource Views Fitur Anggota:

Ditemukan 2 kasus "Chose Descriptive Names (N1)." Penamaan yang kurang deskriptif diperbaiki.

Refactoring pada Fitur Jadwal:

Controller Fitur Jadwal:

Terdapat 1 kasus “Commented-Out Code (C5)” yang dihapuskan dan 1 kasus “Duplication” yang diatasi.

Refactoring pada Fitur Laporan:

Controller Fitur Laporan:

Terdapat 2 kasus “Commented Out Code (C5)” yang dihapuskan dan 1 kasus “Dead Code (G9) yang dihapus.

Refactoring pada Fitur Login:

Controller Fitur Login:

Terdapat 3 kasus “Commented Out Code (C5)” yang dihapuskan.

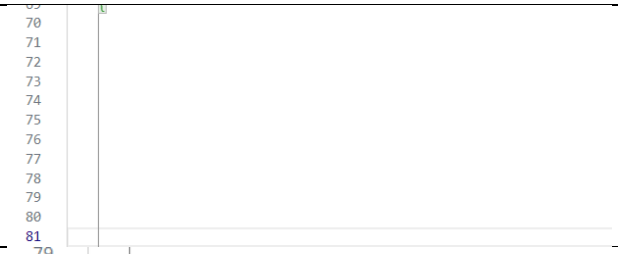
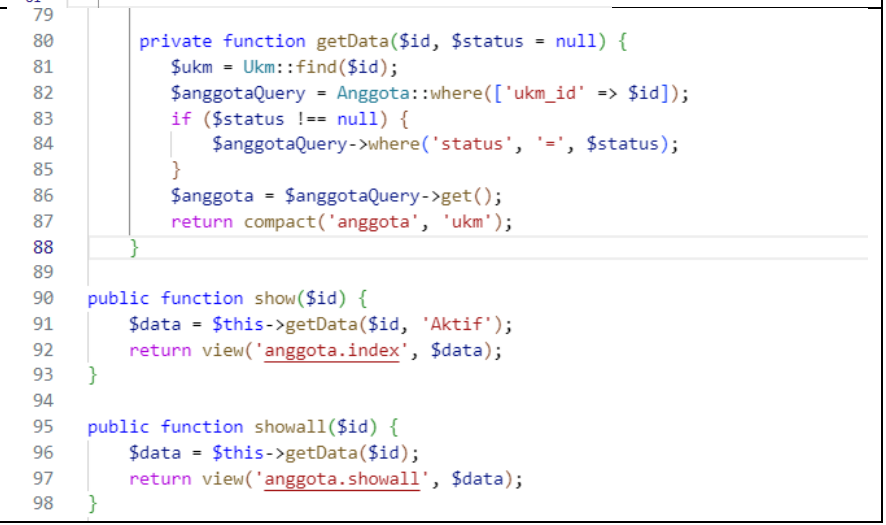
Refactoring Laporan Mahasiswa:

Controller Fitur Laporan Mahasiswa:

Terdapat 3 kasus “Commented Out Code (C5)” yang dihapuskan.

3. 5 Hasil Implementasi *Refactoring*

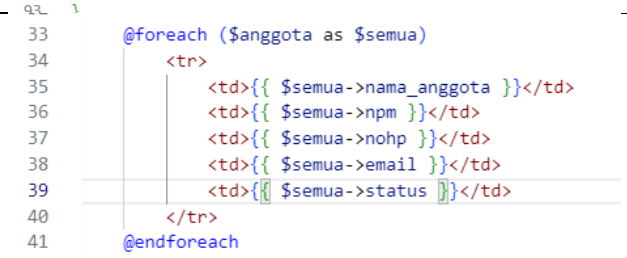
Hasil implementasi refactoring mencakup beberapa contoh perbaikan dari berbagai kategori untuk meningkatkan kualitas dan kebersihan kode.

No.	Smells Code	Code
1	C5	
2	G5	

```
    private function getData($id, $status = null) {
        $ukm = Ukm::find($id);
        $anggotaQuery = Anggota::where(['ukm_id' => $id]);
        if ($status !== null) {
            $anggotaQuery->where('status', '=', $status);
        }
        $anggota = $anggotaQuery->get();
        return compact('anggota', 'ukm');
    }

public function show($id) {
    $data = $this->getData($id, 'Aktif');
    return view('anggota.index', $data);
}

public function showall($id) {
    $data = $this->getData($id);
    return view('anggota.showall', $data);
}
```

3.	G30	
4.	N1	
Total Kategori		4

```
    public function store(Request $request)
{
    $this->validateRequest($request);

    $jadwal = new Jadwal;
    $this->fillJadwalData($jadwal, $request);

    $this->mergeHari($jadwal, $request);

    $jadwal->save();

    return redirect('/jadwal')->with('status', 'Data Jadwal Berhasil Ditambahkan!');
}

private function validateRequest(Request $request)
{
    $request->validate([
        'ukm_id'        => ['required', 'unique:jadwal,ukm_id'],
        'waktu_mulai'   => ['required'],
        'waktu_selesai' => ['required'],
        'hari'          => ['required'],
        'tempat'        => ['required']
    ], [
        'required'      => 'Field harus diisi!',
        'unique'        => 'Jadwal sudah ada!'
    ]);
}
```

```
    @foreach ($anggota as $semua)
        <tr>
            <td>{{ $semua->nama_anggota }}</td>
            <td>{{ $semua->npm }}</td>
            <td>{{ $semua->nohp }}</td>
            <td>{{ $semua->email }}</td>
            <td>{{ $semua->status }}</td>
        </tr>
    @endforeach
```

4. KESIMPULAN

Refactoring pada aplikasi SIMUDAH dilakukan untuk mengatasi berbagai code *smells*, meningkatkan keterbacaan, dan mengurangi kompleksitas. Total 36 *refactoring* dilakukan, membawa perubahan positif dalam kode program dan potensi peningkatan performa aplikasi. Selain itu, perubahan pada jumlah baris kode memberikan gambaran terkait efisiensi dalam pengembangan.

5. DAFTAR PUSTAKA

A.J.Paramita, A. H. E. d. D. K., 2022. Pendeteksi Code Smell Pada Perusahaan Website. *Syntax lit,* 7(10.36418), p. 53.

F.hadiansyah, 2019. SKRIPSI, Hal.1-6, 2019..

Gunawan, D. H. S. d. I., 2003. Studi Penggunaan Visual Studio 6.0 untuk pengembangan SIstem informasi berkelas enterprisw. *J.inform,* Volume 4, pp. 27-34.

I.Liem, H. P. d., 2010. Deteksi Code Smell pada Kode Program dalam Representasi AST dengan pendekatan By rules. Volume 2010, p. 6.

M.idris, F. d., 2021. Implementasi Clean Code pada pegembangan berbasis web. *Automata,* Volume 2 no 2, pp. 113-116.

M.lidnillah, D. d., 2008. Heuristik dalam pemecahan masalah matematika dan pembelajaran di sd.

R.C.Martin, 2016. *Clean Code.* s.l.:10.1007/978-1-4842-1212-7_3.

S.Parera, H. S. d. L. W., 2016. Application of genetic algoritm for class scheduling. *Faculty of science and technology UIN jakarta,* 10.11.09/CITSM.2016.7577525(10.1109), pp. 1-5.

Susetyo, A. d. Y., 2023. Analisis Quality Code Menggunakan Sonarqube Dalam Suatu Aplikasi berbasis laravel. *IT EXPLORE J Penerapan Tekno. Inf dan Komun,* 2(10.242446), pp. 99-103.

T.yulianto, 2017. Pengenalan PHP. *Ilmiu Komput,* pp. 1-9.