

## REFACTORING APLIKASI SALON BLISS SESUAI PRINSIP CLEAN CODE BERBASIS SMELLS DAN HEURISTICS

Nor Hafiza, M. Asep Subandri

Teknik Informatika, Politeknik Negeri Bengkalis  
Jalan Bathin Alam Desa Sungai Alam Kec. Bengkalis, Kab. Bengkalis, Indonesia  
norhafiza24@gmail.com

### ABSTRAK

Kualitas kode merupakan aspek krusial dalam pengembangan perangkat lunak untuk memastikan keterbacaan, efisiensi, dan kemudahan pemeliharaan sistem. Namun, pada aplikasi Salon Bliss berbasis Laravel, ditemukan permasalahan code smell seperti tight coupling, duplicate code, long method, dan complex conditional yang menyulitkan proses pengembangan lanjutan. Oleh karena itu, penelitian ini bertujuan untuk menerapkan prinsip Clean Code melalui pendekatan code smells dan heuristics dalam proses refactoring guna meningkatkan kualitas kode tanpa mengubah fungsionalitas utama aplikasi. Metode penelitian yang digunakan meliputi identifikasi code smell secara manual berdasarkan prinsip Clean Code Robert C. Martin, pemetaan temuan ke heuristics yang relevan, dan pelaksanaan refactoring (seperti penerapan dependency injection, pemisahan tanggung jawab, serta penyederhanaan struktur kondisi). Hasil penelitian membuktikan bahwa refactoring berhasil mengeliminasi code smells secara keseluruhan (pengurangan 100%), menurunkan tingkat ketergantungan antar komponen, serta meningkatkan modularitas kode. Validasi bersama dua orang programmer juga mengonfirmasi bahwa struktur kode pasca-refactoring menjadi lebih terstruktur, mudah dipahami, dan efisien untuk dipelihara.

**Kata kunci :** *Clean Code, Code Smells, Heuristics, Refactoring, Laravel*

### 1. PENDAHULUAN

Dalam proses pengembangan perangkat lunak, penyusunan kode yang rapi dan bersih merupakan tantangan yang sering dihadapi pengembang. Tekanan untuk menyelesaikan proyek dalam waktu singkat sering menyebabkan pengembang mengabaikan praktik penulisan kode yang baik, sehingga memunculkan kode yang rumit, sulit dipahami, tidak terstruktur, serta rentan terhadap kesalahan. Prinsip clean code menekankan bahwa kode harus berfungsi dengan baik sekaligus mudah dipahami oleh berbagai pengembang yang terlibat di dalam proyek [1]. Namun, pada praktiknya masih sering ditemukan struktur yang tidak rapi, penamaan variabel yang tidak jelas, serta fungsi berulang yang dikenal dengan istilah code smells. Kode yang mengandung code smells ini dapat memperlambat proses pemeliharaan, memicu kesalahan yang sulit dilacak, dan menimbulkan kendala efisiensi dalam kolaborasi tim [2].

Penelitian ini berfokus pada aplikasi manajemen Salon Bliss berbasis web yang menggunakan framework Laravel. Aplikasi ini dirancang untuk membantu pengelolaan layanan, pelanggan, jadwal pemesanan, karyawan, serta transaksi secara daring dan waktu nyata. Namun, permasalahan utama yang dijumpai pada source code aplikasi Salon Bliss adalah ditemukannya sejumlah kelemahan desain (code smells) yang cukup dominan. Permasalahan tersebut meliputi tingginya ketergantungan antar komponen (tight coupling) akibat penggunaan pemanggilan statis (static access), adanya pengulangan logika (duplicate code), metode yang terlalu panjang (long method), serta percabangan bersyarat yang rumit (complex conditional). Berbagai permasalahan ini berpotensi

menurunkan kualitas struktur kode secara signifikan dan menyulitkan proses pengembangan lanjutan sistem di masa mendatang [3].

Sebagai rencana penyelesaian masalah, penelitian ini menerapkan prinsip clean code menggunakan pendekatan smells dan heuristics. Metode code smells difungsikan untuk mengidentifikasi kelemahan logika secara manual dan sistematis berdasarkan prinsip Robert C. Martin, di mana peneliti membaca langsung struktur kelas dan method pada program. Setiap temuan code smell tersebut kemudian dipetakan pada heuristics yang relevan sebagai panduan pengambilan keputusan desain perbaikan. Selanjutnya, perbaikan dieksekusi melalui tahap implementasi refactoring (seperti penerapan dependency injection, pemisahan tanggung jawab, dan penyederhanaan kondisi) untuk menuntaskan masalah internal tersebut tanpa mengubah fungsionalitas, output, maupun stabilitas utama dari aplikasi [4].

Meskipun kajian mengenai penerapan clean code dan refactoring telah banyak dilakukan, sebagian besar penelitian terdahulu masih sangat bergantung pada penggunaan tools deteksi otomatis tanpa disertai analisis manual yang mendalam berdasarkan heuristik. Selain itu, belum banyak studi yang mengevaluasi secara kuantitatif perbandingan kualitas perangkat lunak sebelum dan sesudah refactoring [5]. Oleh karena itu, tujuan dari penelitian ini adalah untuk mengintegrasikan tahapan deteksi manual yang terstruktur guna mereduksi permasalahan code smells secara keseluruhan, sekaligus meningkatkan keterbacaan (readability), efisiensi, dan kemudahan

pemeliharaan (maintainability) kode pada aplikasi Salon Bliss pasca-refactoring.

## 2. TINJAUAN PUSTAKA

### 2.1. Penelitian Terdahulu

Penelitian mengenai penerapan prinsip clean code, evaluasi sistem, dan refactoring telah banyak dieksplorasi dalam pengembangan perangkat lunak. Penerapan clean code terbukti dapat secara signifikan mempermudah pemahaman, kolaborasi programmer, serta meningkatkan indeks pemeliharaan (*maintainability index*) sistem informasi [1][2]. Sejalan dengan hal tersebut, upaya mereduksi hingga 330 baris kode tanpa mengubah fitur fungsional aplikasi sukses dicapai melalui pengidentifikasian code smells (seperti penamaan variabel yang tidak deskriptif dan duplikasi kode) yang kemudian diperbaiki menggunakan heuristics yang sesuai [3].

Selain kualitas internal kode, konsistensi pengkodean dan evaluasi antarmuka juga menjadi hal yang krusial dalam merancang perangkat lunak. Pengintegrasian linter ESLint pada continuous integration pipeline untuk mendeteksi pelanggaran style guide secara otomatis dibuktikan mampu mempercepat proses code review hingga 30% [4]. Selanjutnya, evaluasi fungsionalitas dan kelayakan sistem menggunakan metode Heuristic Evaluation juga secara efektif terbukti dapat menurunkan severity rating pada permasalahan antarmuka pengguna, menjadikan sistem lebih terstruktur dan berpusat pada pengalaman pengguna akhir [5][6].

Pada ranah manajemen layanan, pengembangan sistem reservasi salon berbasis web menggunakan framework Laravel terbukti mempermudah proses penjadwalan secara daring serta meminimalkan kesalahan pencatatan data [7]. Untuk mendukung stabilitas pada sistem kompleks semacam itu, refactoring diandalkan sebagai teknik optimasi utama. Pelaksanaan refactoring menggunakan teknik extract method dipastikan sangat efektif dalam mengeliminasi baris kode duplikat sekaligus mengurangi risiko kerentanan bug [8]. Dalam implementasi yang jauh lebih mutakhir, deteksi code smell otomatis (seperti penanganan lazy class) menggunakan metrik perangkat lunak bahkan terbukti mampu menghemat waktu pemeliharaan secara drastis [9][10].

Berdasarkan berbagai kajian terdahulu tersebut, penelitian ini membedakan diri dengan memadukan tahapan evaluasi manual code smells secara komprehensif yang murni berpedoman pada heuristics, sebelum mengimplementasikan restrukturisasi refactoring yang spesifik pada aplikasi manajemen Salon Bliss.

### 2.2. Clean Code

Merupakan pendekatan penulisan kode yang mengutamakan tingkat keterbacaan tinggi, kemudahan pemahaman, serta efisiensi pemeliharaan (maintenance). Konsep ini menuntut kepatuhan terhadap beberapa prinsip utama, seperti penulisan

nama variabel yang representatif (meaningful names), pembuatan fungsi berskala kecil dengan tanggung jawab tunggal (single responsibility), serta menghindari pengulangan alur logika (Don't Repeat Yourself / DRY) [1].

### 2.3. Code Smells

Code smells adalah kelemahan atau indikasi desain struktural yang buruk di dalam baris kode. Meskipun aplikasi tetap berjalan lancar, keberadaan code smells (misalnya long method, magic number, atau pemanggilan kelas secara statis) berpotensi memicu kerentanan bug serta menyulitkan proses penambahan fitur di masa depan [3][10]. Untuk mengidentifikasi serta memperbaiki kelemahan tersebut, pengembang memanfaatkan Heuristics. Heuristics adalah sekumpulan pedoman praktis atau aturan pokok turunan dari prinsip clean code (misalnya penyederhanaan kondisi logika dan pemisahan modul) yang membantu pengambilan keputusan desain sistem secara optimal [9].

### 2.4. Heuristic

Heuristic didefinisikan sebagai pedoman praktis yang mendukung pengembangan kode yang bersih, mudah dipahami, dan efisien, dengan referensi ke prinsip Clean Code Robert C. Martin. Disebutkan bahwa Heuristic, seperti Single Responsibility Principle dan Encapsulation, adalah aturan praktis yang krusial untuk menghadapi kompleksitas pengembangan perangkat lunak. Heuristic membantu pengembang membuat keputusan yang lebih baik dalam menulis, membaca, dan memelihara kode [7].

### 2.5. Refactoring

Proses ini berfokus pada restrukturisasi dan perbaikan bentuk internal perangkat lunak tanpa sedikit pun mengubah fungsionalitas dan output eksternal dari aplikasi tersebut. Dalam praktiknya, penelitian ini dilakukan pada ekosistem PHP, yakni bahasa pemrograman server-side unggulan untuk perancangan aplikasi dinamis, yang ditenagai oleh Laravel sebagai framework utamanya. Framework Laravel menyediakan struktur MVC yang kuat serta berbagai fitur turunan otentikasi yang mempercepat siklus pengembangan. Keseluruhan proses pembacaan dan modifikasi source code ini dieksekusi menggunakan bantuan teks editor Visual Studio Code, yang ringan dan mendukung berbagai integrasi plugin esensial [8].

## 3. METODE PENELITIAN

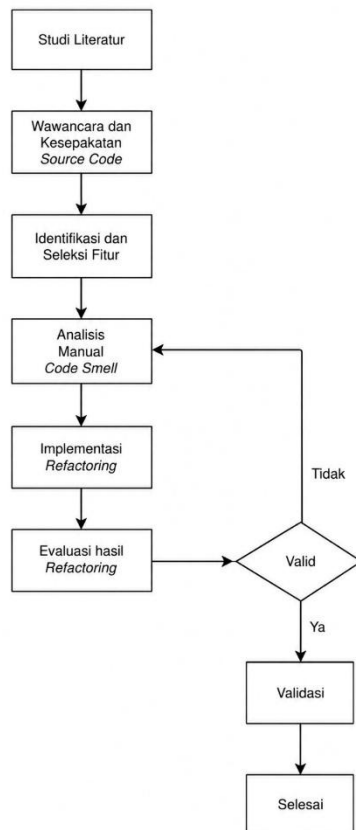
### 3.1. Data dan Alat Penelitian

Penelitian ini menggunakan data utama berupa source code aplikasi Salon Bliss yang dikembangkan menggunakan framework Laravel. Kode program ini menjadi objek analisis utama dalam proses identifikasi code smells dan pengimplementasian prinsip clean code melalui refactoring. Adapun alat yang digunakan untuk menunjang penelitian meliputi perangkat keras

berupa laptop berspesifikasi Intel Core i5 dengan RAM 8 GB, serta perangkat lunak yang mencakup Visual Studio Code sebagai teks editor, basis data MySQL, bahasa pemrograman PHP, dan penjelajah web Google Chrome.

### 3.2. Prosedur Penelitian

Prosedur pelaksanaan penelitian ini mengikuti beberapa tahapan sistematis yang digambarkan pada alur berikut.



Gambar 1. Prosedur Penelitian

Dari Gambar 1, dapat dilihat bahwa prosedur penelitian diawali dengan studi literatur untuk mengkaji konsep terkait clean code, code smells, dan teknik refactoring. Setelah itu, dilakukan kesepakatan terhadap source code proyek dan identifikasi seleksi fitur guna menentukan bagian yang terindikasi memiliki potensi kelemahan desain. Tahap selanjutnya adalah melakukan analisis manual code smells dengan mengevaluasi langsung struktur kelas dan method untuk mendeteksi masalah seperti pemanggilan statis (static access), pengulangan logika (duplicate code), percabangan rumit (complex conditional), dan method yang terlalu panjang (long method). Berdasarkan hasil temuan analisis manual tersebut, perbaikan dieksekusi melalui tahap implementasi refactoring dengan menerapkan heuristics clean code yang sesuai pada setiap masalah yang teridentifikasi.

Tabel 1. Contoh Heuristic

| Code Smell            | Heuristics                             | Keterangan                                                                                                                       |
|-----------------------|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| Long Method           | Simplify Conditionals / Extract Method | Memecah method yang panjang menjadi beberapa fungsi yang lebih kecil agar lebih terfokus, mudah dipahami, dan mudah dipelihara.  |
| Duplicate Code        | Avoid Duplication (DRY)                | Menggabungkan logika atau kode yang berulang ke dalam satu method untuk meningkatkan konsistensi dan mengurangi redundansi.      |
| Magic Number          | Avoid Hardcoding                       | Mengganti nilai numerik yang ditulis langsung dalam kode dengan konstanta atau konfigurasi agar lebih mudah dipahami dan diubah. |
| Tight Coupling        | Reduce Coupling                        | Mengurangi ketergantungan antar komponen dengan menerapkan dependency injection, interface, atau service layer.                  |
| Cyclomatic Complexity | Simplify Conditionals                  | Menyederhanakan struktur percabangan dan logika program sehingga kompleksitas kode menjadi lebih rendah dan mudah dikelola.      |

Dari Tabel 1 dapat dilihat bahwa setiap jenis kelemahan (code smells) memiliki pemetaan solusi heuristics secara spesifik. Heuristics ini berfungsi untuk memandu tahapan implementasi refactoring agar restrukturisasi pada aplikasi Salon Bliss berjalan lebih terarah, konsisten, dan tidak sedikit pun mengubah perilaku fungsional dari aplikasi.

### 3.3. Analisis Kebutuhan Sistem

Dalam menjalankan proses refactoring, diperlukan pemenuhan pedoman fungsional dan non-fungsional sistem. Secara fungsional, kode program harus menggunakan penamaan variabel atau kelas yang representatif, memiliki pemisahan modul dengan tugas tunggal, serta menghindari duplikasi logika agar proses perbaikan kesalahan (debug) menjadi lebih mudah. Secara non-fungsional, hasil akhir kode diharapkan sangat konsisten dalam gaya penamaan, mudah dipahami (readability), dan memiliki tingkat modularitas tinggi sehingga kolaborasi tim dan tahapan ekspansi fitur baru dapat berjalan lebih responsif dan efektif.

### 3.4. Evaluasi dan Validasi Refactoring

Tahap akhir dari alur penelitian berfokus pada evaluasi kondisi pra- dan pasca-refactoring serta pelaksanaan validasi. Penilaian efektivitas diukur menggunakan bantuan PHP Mess Detector (PHPMD) untuk membandingkan penurunan jumlah code smells dan kompleksitas method secara metrik. Selanjutnya, untuk memastikan bahwa fungsionalitas sistem tidak rusak serta melihat keberhasilan peningkatan perbaikan struktur, penelitian ini melakukan validasi secara langsung bersama dua orang programmer profesional yang akan mengevaluasi tingkat keterbacaan, keteraturan kode, serta kemudahan proses pemeliharaan program pasca-refactoring

## 4. HASIL DAN PEMBAHASAN

### 4.1. Hasil Penelitian

Penelitian ini diawali dengan mengidentifikasi keberadaan code smells yang berpotensi menurunkan kualitas kode, khususnya dari sisi keterbacaan dan kemudahan pemeliharaan (maintainability). Identifikasi dilakukan secara manual melalui analisis struktur source code dengan mengacu pada prinsip clean code serta pendekatan smells dan heuristics. Analisis difokuskan pada panjang method, duplikasi logika, penggunaan magic number, serta pemanggilan kelas secara statis (static access) sebelum proses refactoring dilakukan.

Tabel 2. Rekapitulasi Code Smell Pra-Refactoring

| No | Nama Class                   | Jenis Code Smell                                                  | Jumlah |
|----|------------------------------|-------------------------------------------------------------------|--------|
| 1  | AdminDashboardHomeController | Long Method, Duplicate Code, Magic Number                         | 3      |
| 2  | CartController               | Long Method, Duplicate Code                                       | 2      |
| 3  | AddingServiceToCart          | Long Method, Duplicate Code, Cyclomatic Complexity, Static Access | 4      |
| 4  | CustomerServicesView         | Static Access, Long Method, Magic Number                          | 3      |
| 5  | TelescopeServiceProvider     | Static Access, Duplicate Code, Cyclomatic Complexity              | 3      |

Dari Tabel 2, dapat dilihat rekapitulasi sampel dominan dari penemuan code smells pada kode aplikasi Salon Bliss sebelum dilakukan perbaikan. Jenis kelemahan utama yang paling sering muncul meliputi Static Access, Duplicate Code, Magic Number, dan Long Method yang tersebar di berbagai berkas seperti Controller, Livewire, Middleware, dan Providers. Tingginya kemunculan Static Access dan Duplicate Code ini mengindikasikan bahwa struktur kode masih memiliki ketergantungan yang sangat tinggi (tight coupling) dan duplikasi logika yang berulang.

### 4.2. Implementasi Refactoring

Tahapan implementasi refactoring dieksekusi pada komponen Controller, Livewire, Middleware, dan Jobs dengan memetakan setiap temuan smells pada pemecahan heuristics yang tepat. Pada penanganan Long Method dan Cyclomatic Complexity, baris fungsi yang memuat terlalu banyak tanggung jawab dipisahkan menjadi beberapa method berukuran kecil (Extract Method / Single Responsibility). Untuk kasus Duplicate Code, logika pencarian yang berulang digabungkan menggunakan aturan Don't Repeat Yourself (DRY). Sedangkan pada kelemahan Magic Number (seperti angka batas penomoran tanpa keterangan), sistem diperbaiki dengan menerapkan variabel konstanta pembantu. Terakhir, Static Access yang memicu tight coupling diselesaikan melalui transisi pengadopsian dependency injection menggunakan objek model.

### 4.3. Evaluasi Kualitas Kode

Evaluasi pra dan pasca-refactoring dilakukan untuk mengukur efektivitas penerapan prinsip clean code. Penilaian ini membandingkan hasil identifikasi sebelum dan sesudah modifikasi menggunakan PHP Mess Detector (PHPMD).

Tabel 3. Hasil Evaluasi Kode Refactoring

| No | Metrik                | Pra-Refactoring | Pasca-Refactoring | Perubahan (%) |
|----|-----------------------|-----------------|-------------------|---------------|
| 1  | Long Method           | 8               | 0                 | 100%          |
| 2  | Duplicate Code        | 8               | 0                 | 100%          |
| 3  | Magic Number          | 12              | 0                 | 100%          |
| 4  | Static Access         | 13              | 0                 | 100%          |
| 5  | Cyclomatic Complexity | 2               | 0                 | 100%          |
|    | <b>Total</b>          | <b>43</b>       | <b>0</b>          | <b>100%</b>   |

Dari Tabel 3, dapat dilihat bahwa seluruh indikator code smells yang terdeteksi pada tahap evaluasi pra-refactoring (total 43 temuan dominan) berhasil direduksi secara penuh menjadi 0 pasca-refactoring. Hal ini menunjukkan bahwa penerapan refactoring menggunakan heuristics clean code terbukti memberikan dampak positif dalam penyederhanaan kompleksitas struktural sebesar 100%.

### 4.4. Validasi Bersama Programmer

Selain metrik kuantitatif, penelitian ini melakukan validasi kualitatif secara langsung bersama dua orang programmer profesional. Para evaluator diinstruksikan untuk membandingkan cuplikan kode sebelum dan sesudah perbaikan, kemudian memberikan pendapat terkait aspek keterbacaan, keteraturan struktur, dan kemudahan pengembangan lebih lanjut.

Tabel 4. Hasil Validasi Bersama Programmer

| No | Bagian yang Dinilai    | Programmer 1                                                                          | Programmer 2                                                   |
|----|------------------------|---------------------------------------------------------------------------------------|----------------------------------------------------------------|
| 1  | Keterbacaan Kode       | Lebih mudah dipahami karena fungsi dipisahkan menjadi bagian-bagian yang lebih kecil. | Lebih mudah dipahami karena struktur kode menjadi lebih jelas. |
| 2  | Kemudahan Pengembangan | Perubahan dan pemeliharaan kode lebih mudah dilakukan.                                | Kode menjadi lebih mudah untuk dikembangkan lebih lanjut.      |
| 3  | Struktur Kode          | Struktur kode menjadi lebih rapi dan terorganisir.                                    | Struktur kode menjadi lebih teratur dan sistematis.            |
| 4  | Code Smell             | Jumlah code smell berkurang setelah refactoring.                                      | Terjadi penurunan jumlah code smell pada kode program.         |

Dari Tabel 4, dapat dilihat bahwa hasil ulasan dari kedua programmer mengonfirmasi perbaikan kualitas. Fungsi dan struktur di dalam source code dinilai menjadi jauh lebih rapi, terstruktur, serta secara signifikan menekan angka kompleksitas sehingga sangat efisien untuk proses pemeliharaan.



Gambar 2. Dokumentasi validasi programmer

Dari Gambar 2, dapat dilihat proses validasi hasil refactoring yang dilaksanakan bersama dua orang programmer untuk meninjau secara langsung kualitas perbaikan skrip pada aplikasi.

#### 4.5. Pembahasan Hasil

Berdasarkan hasil identifikasi manual melalui pembacaan struktur kode yang merujuk pada prinsip clean code, dijumpai beberapa permasalahan dominan, terutama Tight Coupling akibat masifnya penggunaan Static Access pada pemanggilan model dan job, serta Duplicate Code. Tumpukan permasalahan tersebut menyebabkan rendahnya fleksibilitas perangkat lunak, melonjaknya ketergantungan antar komponen, serta menyulitkan proses peninjauan sistem oleh pengembang baru.

Proses perbaikan mengatasi kelemahan tersebut dengan mengeksekusi heuristics refactoring spesifik. Penggunaan pola Dependency Injection sangat sukses menekan persentase Tight Coupling. Pemusatan fungsi turunan juga ampuh meratakan duplikasi kode, diiringi perbaikan alur kontrol percabangan untuk menstabilkan kompleksitas siklomatik program.

Hasil evaluasi kuantitatif dan kualitatif secara komprehensif membuktikan bahwa refactoring ini sukses menciptakan struktur perangkat lunak yang lebih modular, konsisten, dan mudah dipahami, tanpa sedikit pun mencederai atau mengubah fungsionalitas fisis dari antarmuka aplikasi Salon Bliss. Keberhasilan pengadopsian standar internal kode ini dikonfirmasi efektif dalam meningkatkan readability, maintainability, dan testability produk secara jangka panjang.

#### 4.6. Pengujian Aplikasi

Untuk mengukur keberhasilan proses refactoring yang telah diterapkan pada aplikasi Salon Bliss, penelitian ini melakukan pengujian aplikasi yang berfokus pada dua metrik utama, yakni pengujian kualitas struktur kode secara kuantitatif dan pengujian fungsionalitas perangkat lunak. Pengujian kualitas source code dilakukan menggunakan alat bantu PHP Mess Detector (PHPMD) untuk membandingkan kondisi program pra dan pasca-refactoring. Hasil pengujian metrik menunjukkan bahwa seluruh indikator kelemahan kode (seperti Long Method, Duplicate Code, Magic Number, Static Access, dan Cyclomatic Complexity) berhasil direduksi secara penuh dari total 43 temuan dominan menjadi 0 temuan (penurunan kompleksitas sebesar 100%).

Selanjutnya, pengujian fungsionalitas dilakukan untuk memastikan bahwa restrukturisasi internal kode tidak merusak perilaku eksternal sistem. Pengujian dan validasi fungsional ini diuji secara langsung dengan melibatkan dua orang programmer profesional. Hasil pengujian membuktikan bahwa seluruh fitur utama aplikasi—seperti fitur keranjang, reservasi pelanggan, serta pengelolaan dasbor admin—tetap berjalan stabil dan normal tanpa ada sedikit pun perubahan pada output sistem aslinya. Selain fungsionalitas yang terbukti tetap terjaga dengan baik, performa pengembangan lanjutan aplikasi dinilai sukses menjadi lebih terstruktur, modular, dan efisien untuk dipelihara.

## 5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian, identifikasi awal pada aplikasi Salon Bliss menemukan sejumlah permasalahan berupa code smell dominan seperti Static Access, Duplicate Code, Long Method, dan Cyclomatic Complexity pada bagian controller dan service, yang berdampak pada tingginya ketergantungan antar komponen (tight coupling) serta rendahnya kualitas internal kode. Penerapan proses refactoring dengan memanfaatkan pedoman heuristics Clean Code terbukti berhasil mereduksi tingkat permasalahan tersebut secara keseluruhan, sehingga struktur program menjadi jauh lebih rapi, modular, terarah, dan mudah dipelihara. Hasil validasi kualitatif bersama praktisi programmer juga mengonfirmasi bahwa perbaikan ini sukses memperjelas struktur kode serta memperlancar proses pengembangan perangkat lunak tanpa sedikit pun mengubah fungsi utama dari sistem aslinya. Sebagai tindak lanjut perbaikan keberlanjutan, pengembang disarankan untuk secara konsisten menerapkan prinsip Clean Code dan melakukan refactoring berkala guna menjaga kualitas kode aplikasi. Untuk penelitian selanjutnya, sangat direkomendasikan untuk memanfaatkan tools analisis kode yang lebih komprehensif seperti SonarQube, PHPStan, atau Psalm, serta melibatkan jumlah evaluator programmer yang lebih banyak agar evaluasi efisiensi perangkat lunak dapat dianalisis dari perspektif yang lebih mendalam dan luas.

## DAFTAR PUSTAKA

- [1] "Implementation of Clean Code Based on Smells and Heuristics MDP SIGN Application.pdf."
- [2] P. D. Yuningsih and L. U. Utami, "Sistem Informasi Online Booking Berbasis Web Pada Pheo Studi Salon," *J. Tekinfo*, vol. 18, no. 1, pp. 193–200, 2024, [Online]. Available: <https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index>
- [3] M. Irfan, W. Widhiarso, and M. Rachmadi, "Penerapan Clean Code Pada Aplikasi Siizin Berdasarkan Smells And Heuristics," 3rd MDP Student Conf. 2024, pp. 95–102, 2024.
- [4] H. Fredianto, "Optimalisasi Perangkat Lunak Menggunakan Metode Refactoring," 2021. doi: 10.46799/jsa.v2i10.326.
- [5] M. Inayah, F.I., dan Idris, "Implementasi Clean Code pada Pengembangan Berbasis Web," *Automata*, vol. 2, no. 2, pp. 113–116, 2021.
- [6] A. P. Anggreini, "Pengembangan Sistem Presensi Karyawan Dengan Face Recognition Menggunakan Model Facenet," p. 68, 2024, [Online]. Available: <https://dspace.uui.ac.id/handle/123456789/51600>
- [7] C. Sanora and H. Widagdo, "Analisis Loyalitas Pelanggan Pada Bliss Salon Palembang 2022-2023," 2023. doi: 10.35957/prmm.v4i2.4188.
- [8] "View Implementation of Clean Code Based on Smells and Heuristics MDP SIGN Application.pdf."
- [9] F. K. S. Dewi, Yonathan Dri Handarkho, and Felcia Veronica Prasetyo, "Analisis Usability Menggunakan Metode Heuristic Evaluation dan Web Usability Evaluation Tool pada Website ACC Career," *J. Buana Inform.*, vol. 13, no. 02, pp. 126–135, 2022, doi: 10.24002/jbi.v13i02.6488.
- [10] M. P. Zia, D. S. Kusumo, and ..., "Konsistensi Kode Pada Bahasa Pemrograman Javascript Menggunakan Linter Pada Continuous Integration Pipeline," *eProceedings ...*, vol. 8, no. 2, pp. 3245–3252, 2021.
- [11] "Tampilan Evaluasi Usability dengan Menerapkan Metode Heuristic Evaluation pada Website Dinas Pendidikan Kota Batu.pdf."
- [12] "View of Penerapan Metode Heuristic Evaluation Untuk Evaluasi User Interface Aplikasi Lazada.pdf."
- [13] B. J. M. Putra, Ardyani Purwatama, and P. O. D. A. Purnamasari, "Pengembangan Aplikasi Reservasi Salon Keshoo Berbasis Web," *J. Electr. Electron. Mech. Inform. Soc. Appl. Sci.*, vol. 2, no. 2, pp. 24–30, 2023, doi: 10.58991/eemisas.v2i2.42.
- [14] M. Irfan, "Model Analisis Aplikasi Siizin Menggunakan Clean Code Berdasarkan Smells And Heuristics Oleh : Fakultas Ilmu Komputer Dan Rekayasa Uniersitas Multi Data Palembang Fakultas Ilmu Komputer dan Rekayasa Universitas Multi Data Palembang," 2024.
- [15] "View of Optimalisasi Perangkat Lunak Menggunakan Metode Refactoring.pdf."
- [16] "Tampilan Tool Refactoring Otomatis untuk Menangani Lazy Class Code Smell dengan Pendekatan Software Metrics.pdf."
- [17] "View of Implementasi Clean Code Pada Pengembangan Aplikasi Berbasis Web.pdf."
- [18] E. Kusuma, "Deteksi Code Smell dengan Pendekatan Machine Learning: Analisis Bibliometrik," 2024. doi: 10.32877/bt.v7i2.1822.
- [19] M. Amazona, "Oleh : Fakultas Ilmu Komputer Dan Rekayasa Universitas Multi Data Palembang Palembang Fakultas Ilmu Komputer dan Rekayasa Universitas Multi Data Palembang," 2024.
- [20] H. Pembahasan and P. Refactoring, "Jurnal : Refactoring Kode dalam Pengembangan Perangkat Lunak," no. 1, pp. 1–5, 2025.
- [21] "Application System Design and Implementation of a Web-Based Bus and Truck Spare Parts Sales Using the Laravel Framework.pdf."
- [22] "Rancang Bangun Sistem Informasi Penerimaan Peserta Didik Baru (PPDB) Sekolah Menengah Kejuruan Telkom 2 Medan Menggunakan Codeigniter.pdf."
- [23] K. Surya Ningsih, N. Jamilah Aruan, and A. Taufik Al Afkari Siahaan, "Aplikasi Buku Tamu Menggunakan Fitur Kamera Dan Ajax Berbasis Website Pada Kantor Disporakota Medan," 2022.