JKT Jurnal Komputer Terapan
ISSN : 2443-4159 (Print) ISSN: 2460-5255 (Online)
Politeknik Caltex Riau
Empower You to Global Competition

PENGUJIAN KUALITAS KODE PROGRAM APLIKASI BANK SAMPAH DLHK KOTA PEKANBARU MENGGUNAKAN CODE SMELL TOOLS

Mardhiah Fadhli[1], Yuli Fitrisia[2]*, Dini Nurmalasari[3]
Prodi Teknologi Rekayasa Komputer (Politeknik Caltex Riau, Pekanbaru, 28265, Indonesia)[1,2,3]
mardhiah@pcr.ac.id[1], uli@pcr.ac.id[2], dini@pcr.ac.id[3]

**Penulis Koresponden*

ABSTRAK

Kualitas dari kode program akan mempengaruhi kemampuan perangkat lunak untuk dapat mudah dimodifikasi dan dikembangkan serta dipelihara. Code smells merupakan suatu karakteristik dari perangkat lunak yang mengindikasikan permasalahan pada kode dan desain perangkat lunak yang mengakibatkan perangkat lunak sulit untuk dikembangkan dan dilakukan pemeliharaan. Deteksi code smell perlu dilakukan agar dalam pengembangan kedepannya aplikasi dapat lebih mudah dimodifikasi dan dikembangkan. Deteksi code smell dalam sebuah aplikasi dapat membantu programmer untuk mengidentifikasi adanya rancangan kode program yang dapat menyulitkan kedepannya untuk dilakukan modifikasi dan pengembangan. Pendeteksian Code Smell pada aplikasi Bank Sampah DLHK Kota Pekanbaru dilakukan karena adanya permintaan kebutuhan untuk perbaikan dan penambahan fitur dari Bank Sampah DLHK Kota Pekanbaru. Permintaan perbaikan dan penambahan fitur pada aplikasi Bank Sampah DLHK Kota Pekanbaru dilakukan berdasarkan hasil evaluasi aplikasi sebelumnya yang dilakukan oleh pihak DLHK Kota Pekanbaru kepada 10 Bank Sampah Unit dan 25 Nasabah. Berdasarkan hasil evaluasi dan uji coba aplikasi, maka perlu dilakukan penyesuaian karena adanya ketidaksinkronisasian proses dengan mekanisme yang sedang berjalan dimasyarakat terhadap fitur pada aplikasi tersebut. Untuk memudahkan proses modifikasi program maka pendeteksian code smell pada aplikasi yang sudah ada perlu dilakukan, agar programmer dapat menjaga kualitas kode program menjadi lebih mudah untuk dikembangkan. Deteksi Code Smell dilakukan dengan menggunakan tool SonarQube. Hasil dari pengukuran dari dua aplikasi Bank Sampah adalah, pada aplikasi Basada berbasis mobile terdapat 126 Code smells dengan estimasi waktu perbaikan sekitar 2 jam 34 menit. Sedangkan pada aplikasi Basada berbasis website terdeteksi 25 Code smells dengan estimasi waktu perbaikan sekitar 25 menit.

Kata kunci: *Deteksi, Code Smell, Aplikasi Bank Sampah, SonarQube*

ABSTRACT

The quality of the program code will affect the ability of the software to be easily modified, developed and maintained. Code Smells are a characteristic of software that indicates problems with the code and system design, making the software difficult to develop and maintain. Code smell detection needs to be done so that in future development the application can be more easily modified and developed. Detecting code smells in an application can help programmers identify program code designs that can make it difficult to modify and develop in the future. Code Smell detection in the Pekanbaru City DLHK Waste Bank application was carried out due to requests for improvements and additional features from the Pekanbaru City DLHK Waste Bank. Requests for improvements and additional features to the Pekanbaru City DLHK Waste Bank application were made based on the results of previous application evaluations carried out by the Pekanbaru City DLHK for 10 Unit Waste Banks and 25 customers. Based on the results of evaluation and application testing, adjustments need to be made due to the process not being synchronized with the mechanisms currently running in society regarding the features of the application. To facilitate the program modification process, it is necessary to detect code smells in existing applications, so that programmers can maintain the quality of program code and make it easier

to develop. Code Smell Detection is carried out using the SonarQube tool. The results of measurements from the two Waste Bank applications are that in the Mobile Basada application there are 126 code odors with an estimated repair time of around 2 hours 34 minutes. Meanwhile, on the Based Waste Bank website application, 25 odor codes were detected with an estimated repair time of around 25 minutes.

Keywords: *Detection, Code Smell, Waste Bank Application, SonarQube*

Histori Artikel:
Diserahkan: 24 November 2023 Diterima setelah Revisi: 18 July 2024 Diterbitkan: 26 July 2024

1. PENDAHULUAN

Pengukuran kualitas perangkat lunak merupakan hal yang sangat penting dalam suatu pengembangan perangkat lunak. Dalam menjamin kualitas perangkat lunak maka perlu dilakukan pengukuran terhadap perangkat lunak tersebut. Melalui pengukuran, maka akan diperoleh tingkat pencapaian kualitas didalam proyek perangkat lunak yang sedang diamati[1]. Sering kali programmer tidak memperhatikan kaidah bahasa pemrograman yang baik dalam membangun suatu perangkat lunak disebabkan karena terlalu focus untuk mencapai hasil fungsionalitas yang sesuai dengan requirement awal, sehingga dapat menghasilkan code smells. Hal ini dapat menyebabkan kemungkinan implementasi yang buruk dan ketidaksempurnaan kode program. Oleh karena itu perlu dilakukan inspeksi terhadap kode program yang dirancang secara otomatis agar dapat dilakukan perbaikan dalam menghasilkan kode program dengan rancangan yang baik[2].

Pendeteksian code smells menunjukkan masalah kualitas perangkat lunak lebih dalam, sehingga dapat memberikan wawasan untuk perlu mengetahui area kode program yang perlu ditingkatkan[3]. Code smells didefinisikan sebagai penerapan dan implementasi kode program yang buruk yang dapat menyebabkan penurunan kualitas kode program[4]. Adanya code smells juga mengindikasikan permasalahan pada kode program dan desain perangkat lunak sehingga mengakibatkan perangkat lunak sulit untuk dikembangkan dan dilakukan pemeliharaan[5]. Ketika perangkat lunak dimodifikasi, dikembangkan dan diadaptasi terhadap spesifikasi yang baru, sebuah kode program yang dihasilkan juga semakin kompleks dan rumit. Terjadinya peningkatan kode program dalam rancangan perangkat lunak memungkinkan terjadinya error sehingga berdampak pada keamanan suatu perangkat lunak itu sendiri[6]. Ketika terdapat code smells yang terdeteksi pada kode program, akan mempengaruhi sistem jangka panjang, yang artinya perangkat lunak tidak mencapai kondisi optimal dalam hal kinerja dan pemeliharaan. Peningkatan biaya pemeliharaan perangkat lunak umumnya dikaitkan dengan kualitas perangkat lunak yang buruk dan implementasi kode program yang tidak sesuai standar[7].

Pendeteksian Code Smell pada aplikasi Bank Sampah DLHK Kota Pekanbaru dilakukan karena adanya permintaan kebutuhan untuk perbaikan dan penambahan fitur dari Bank Sampah DLHK Kota Pekanbaru. Permintaan perbaikan dan penambahan fitur pada aplikasi Bank Sampah DLHK Kota Pekanbaru dilakukan berdasarkan hasil evaluasi aplikasi sebelumnya yang dilakukan oleh pihak DLHK Kota Pekanbaru kepada 10 Bank Sampah Unit dan 25 Nasabah. Berdasarkan hasil evaluasi tersebut perlu dilakukan perbaikan pada fitur pemesanan dan pencatatan transaksi sampah. Untuk menghasilkan kode program yang berkualitas tentu diperlukan pengukuran terhadap kualitas kode program saat ini. Deteksi code smell perlu dilakukan agar dalam pengembangan kedepannya aplikasi dapat lebih mudah dimodifikasi dan dikembangkan (maintainability). Untuk memudahkan proses modifikasi program maka pendeteksian code smell pada apikasi yang sudah ada perlu dilakukan, agar programmer dapat menjaga kualitas kode program menjadi lebih mudah untuk dikembangkan.

2. TINJAUAN PUSTAKA

Deteksi code smell dalam sebuah aplikasi dapat membantu programmer untuk mengidentifikasi adanya rancangan kode program yang buruk, sehingga dapat menyulitkan untuk dilakukan modifikasi dan pengembangan terhadap kode program[8]. Pendeteksian code smell sendiri dapat dilakukan secara manual oleh manusia ataupun secara otomatis oleh suatu tools. Pendeteksian secara manual akan memerlukan waktu yang cukup lama apalagi jika dilakukan untuk mendeteksi ratusan kode program. Sehingga dibutuhkan sebuah alat yang dapat mengecek kualitas kode program dengan cara otomatis.

Terdapat Tools yang digunakan untuk mendeteksi Code Smell diantaranya CheckStyle, PMD, JDeodorant, Dead Code Detector, in Fusion,iPlasma, dan Stench Blossom[9][2]. Tools lainnya adalah SonarQube. SonarQube adalah alat peninjauan kode otomatis untuk mendeteksi terdapatnya bug, kerentanan, dan code smells didalam kode yang dibuat. SonarQube dapat memeriksa kode program dengan banyak inputan project/aplikasi dalam satu alur kerja[10]. Masing-masing tools memiliki kemampuan yang berbeda untuk mendeteksi code smell.

Penelitian terkait deteksi code smell ini juga merupakan bagian dari keilmuan rekayasa perangkat lunak khususnya pada penjaminan kualitas perangkat lunak dari sisi internal quality. Hasil dari penelitian ini nantinya dapat dijadikan sebagai contoh kasus dalam mata kuliah kualitas perangkat lunak dengan melakukan pengukuran kualitas internal pada kode program. Bagi programmer hasil deteksi code smell dapat dijadikan sebagai langkah untuk melakukan perbaikan pada stuktur kode programnya agar aplikasi menjadi lebih mudah untuk di modifikasi dan dikembangkan.

Penelitian untuk deteksi code smell sudah dilakukan pada aplikasi berbasis java yaitu pada studi kasus pembuatan e-commerce[2]. Evaluasi dilakukan dengan menggunakan tools PMD, Checkstyle, JDeodorant, dan Dead Code Detector. Tools ini dapat digunakan untuk inspeksi secara otomatis dengan kemampuan yang berbeda-beda. Setelah pendeteksian dilakukan terhadap kode program untuk menemukan code smell.

Penelitian serupa juga dilakukan untuk melakukan deteksi code smell seperti Large Class, Big Class, God Class, Lazy Class, Long Method, Brain Method dan God Method untuk melakukan pemrograman dalam tim[11]. Framework tersebut mampu memberikan penjelasan dan sebab-sebab yang rasional ketika framework tersebut mengklasifikasikan bagian dari soure code sebagai code smell.

Penelitian selanjutnya adalah terkait membuat aplikasi untuk mendeteksi code smell berdasarkan metrik feature envy[6], hasil dari penelitian ini adalah metrik feature envy mampu memberikan informasi terkait class dan method yang terindikasi memiliki nilai feature envy. Hasil perhitungan metrik ditampilkan dalam bentuk tabel supaya mempermudah dalam mengelompokkan class dan method tersebut.

Pembangunan aplikasi untuk mendeteksi code smell juga dilakukan oleh Firdaus et al., 2018[5], dalam penelitian tersebut dikembangkan pendeteksian code smell pada tahap pengembangan perangkat lunak khususnya pada tahap perancangan dengan menggunakan class diagram. Pendeteksian dilakukan dengan melakukan parsing pada class diagram. Deteksi code smell juga dilakukan dalam deteksi code smell pada kode program dalam representasi AST dengan Pendekatan *By Rules*[12] yang dilakukan pada kode program hasil praktikum tugas kuliah pemrograman. Hasil dari penelitian ini adalah program detektor mampu mendeteksi 26 smell dengan format .XML.

Code smell merupakan istilah yang menandakan bahwa di suatu kode program mungkin terdapat suatu masalah. Code Smell dapat menyebabkan masalah pemeliharaan dan pemahaman pada proyek perangkat lunak. Code Smell bukanlah bug, hanya dapat mempersulit pengembang

perangkat lunak untuk memahami kode program. Code smell dapat menyebabkan kesulitan untuk memperbaiki dan memutakhirkan kode program untuk pengembang dan pengelola perangkat lunak[9].

2.1 SonarQube Tool

SonarQube adalah salah satu alat untuk melakukan pengecekan kode statis yang open source yang paling banyak digunakan saat ini[13]. SonarQube juga mendefenisikan seperangkat aturan yang menentukan standar pengkodean, sehingga dapat membantu tim pengembang untuk praktik pengkodean yang tepat. Dalam hal ini tim pengembang ingin memahami kualitas kode program yang dibuat, maka SonarQube menghasilkan laporan terkait jenis masalah yang terdeteksi atau aturan pengkodean yang dilanggar. SonarQube memuat beberapa metrik untuk mengidentifikasi kualitas kode program antara lain bugs, vurnerability, security hotspots, maintainability dan duplicate code[14].

Beberapa fitur utama dari SonarQube termasuk:

1. Pemeriksaan Kode Statis: SonarQube melakukan pemeriksaan kode sumber secara statis untuk mengidentifikasi masalah potensial seperti bug, kode yang sulit dipahami, atau pelanggaran aturan penulisan kode.
2. Pemantauan Kualitas Kode: SonarQube dapat digunakan untuk melacak perubahan kualitas kode seiring waktu dan mengukur kemajuan dalam memperbaiki masalah kode.
3. Integrasi dengan Berbagai Bahasa Pemrograman: SonarQube mendukung banyak bahasa pemrograman termasuk Java, C#, JavaScript, Python.
4. Integrasi dengan Berbagai Alat Pengembangan: SonarQube dapat diintegrasikan dengan alat pengembangan seperti Jenkins, Git, dan IDE (Integrated Development Environment) lainnya.
5. Pelaporan dan Visualisasi: SonarQube menyediakan laporan dan visualisasi yang membantu pengembang dan tim pengembangan untuk memahami dan memprioritaskan masalah kode.
6. Penilaian kode: Dengan bantuan skor seperti “A”, “B”, atau “C”, Sonarqube memberikan gambaran tentang kualitas kode dan area yang memerlukan perhatian khusus.

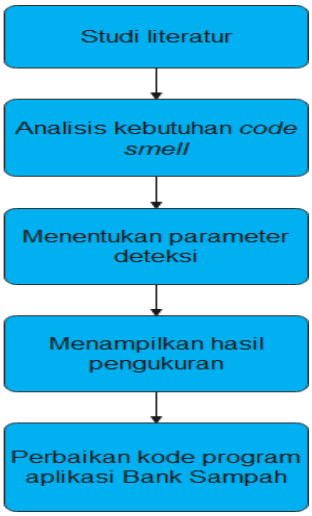
2.2 Code Smells

Code smells adalah karakteristik dari kode program yang dikelompokkan dalam sebuah issues yang dapat menimbulkan permasalahan dikemudian hari[15]. Walaupun code smell bukanlah sebuah bug yang dapat mempengaruhi fungsionalitas system, tetapi menggambarkan kelemahan dari sebuah kode desain. Berikut adalah beberapa contoh code smell yang dapat menimbulkan permasalahan yaitu[16][17][18]:

1. Large Class dan Long Method berisi sebuah class dan method yang melakukan banyak hal sehingga sulit dipahami.
2. Duplicate code yaitu potongan kode program yang sama tetapi berulang dilokasi lain pada kode program aplikasi.
3. God Object yaitu sebuah class atau program yang menjadi pusat system sehingga menjadi sulit untuk dirubah dan beresiko.
4. Dead code yaitu kode yang sudah tidak digunakan lagi tetapi masih ada di program.
5. Feature envy yaitu sebuah method yang lebih tertarik pada bukan class yang seharusnya dia butuhkan.
6. Lazy class yaitu class yang tidak banyak melakukan sesuatu yang dibutuhkan dan sebaiknya class tersebut dieliminasi.
7. Long parameter list yaitu method yang memiliki banyak parameter dapat menjadi masalah dan sulit digunakan.

3. METODE PENELITIAN

Metode penelitian ini terdiri dari beberapa tahapan, mulai dari melakukan studi literatur, menganalisis deteksi code smell, menentukan parameter deteksi, melakukan proses deteksi, menampilkan hasil pengukuran dan selanjutnya melakukan modifikasi program berdasarkan perubahan fitur yang diminta oleh DLHK Kota Pekanbaru. Pada gambar 1 berikut menunjukkan tahapan penelitian.



Gambar 1. Metode Penelitian

Tahapan pertama melakukan studi literatur dengan melakukan pencarian referensi literatur dan kajian pustaka agar mendapatkan informasi yang berhubungan dengan penelitian yang sedang dikerjakan. Tools yang digunakan dalam pembangunan sistem aplikasi deteksi code smell antara lain CheckStyle, PMD, JDeodorant, Dead Code Detector, in Fusion, iPlasma, Stench Blossom dan SonarQube.

Tahapan kedua melakukan analisis kebutuhan code smell. Terdapat banyak sekali code smell yang muncul di lingkungan pengembangan perangkat lunak seperti Large Class, God Class, Lazy Class, Long Method, dan feature envy. Dari beragam code smell yang ada tersebut, cara yang dilakukan untuk mendeteksi keberadaanya juga beragam sehingga perlu dilakukan analisis terlebih dahulu. Pada tahap analisis dilakukan dengan mendefinisikan suatu kode program. Sebuah kode program dapat dituliskan dalam berbagai bahasa pemrograman tertentu. Setiap bahasa memiliki kode program dengan sintaks yang berbeda-beda untuk merepresentasikan satu program yang sama. Deteksi yang dilakukan langsung pada kode program akan bergantung pada bahasa yang digunakan. Namun secara umum struktur dari kode program tersebut akan sama jika dilihat dari model AST (Abstract Syntaxt Tree).

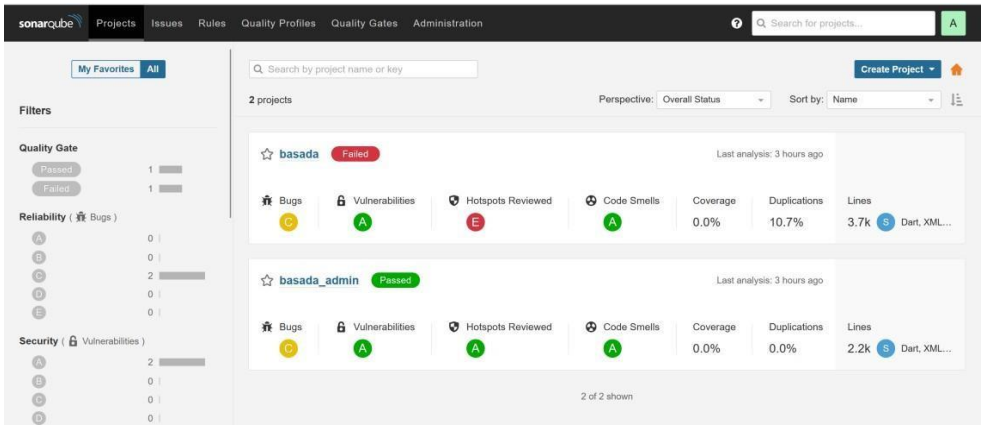
Tahapan ketiga adalah menentukan parameter deteksi. Setiap code smell memiliki kriteria yang menjelaskan kode seperti apa yang mengandung smell. Aturan-aturan yang menandakan code smell beserta kode program yang akan dideteksi disediakan oleh pengguna sebagai parameter masukan untuk perangkat detektornya. Beberapa parameter code smell yaitu Long Parameter List, Large Class, Lazy Class, Feature Envy, Long Method, Dead Code, dan lain-lain. Kriteria sebuah program terdeteksi sebagai smell berdasarkan jenis atau tipe dari code smell yang dimiliki oleh tools yang akan digunakan sebagai detector.

Tahapan keempat yaitu melakukan proses deteksi. Proses deteksi dilakukan dengan menggunakan beberapa tools antara lain CheckStyle, Jdeodorant, Dead Code dan SonarQube. Hal ini dilakukan karena masing-masing tools memiliki kemampuan yang berbeda untuk mendeteksi code smell.

Setelah proses deteksi berhasil dilakukan, maka selanjutnya melakukan pengukuran kualitas dengan menggunakan tools. Hasil pengukuran yang akan dianalisis sangat bergantung dari hasil deteksi code smell. Hasil pengukuran akan dijadikan sebagai dasar untuk melakukan perbaikan pada kode program yang terdeteksi memiliki code smell. Selanjutnya hasil dari perbaikan akan dilakukan pengukuran kembali untuk dapat mengetahui kualitas kode program sudah memenuhi standar yang diharapkan. Setelah kualitas kode program dinyatakan memenuhi standar, maka programmer dapat melakukan modifikasi pada kode program untuk mengakomodasi adanya permintaan perubahan pada fitur di aplikasi Bank Sampah DLHK Kota Pekanbaru.

4. HASIL DAN PEMBAHASAN

SonarQube adalah platform open-source yang dikembangkan oleh SonarSource, yang digunakan untuk inspeksi berkelanjutan pada kualitas kode program. Kemampuan SonarQube dapat menyediakan laporan lebih detail terkait bug, code smells, kerentanan hingga duplikasi kode. Berikut adalah hasil deteksi code smells pada dua project aplikasi bank sampah DLHK Kota Pekabaru pada gambar 2 berikut:



Gambar 2. Hasil deteksi code smells pada halaman dashboard SonarQube

Pendeteksian code smells pada dua project aplikasi, yaitu pada aplikasi basada berbasis mobile dan aplikasi basada berbasis website. Aplikasi basada berbasis mobile adalah aplikasi yang digunakan oleh nasabah bank sampah DLHK Kota Pekanbaru untuk melakukan transaksi menabung sampah dan melihat riwayat transaksi dan saldo nasabah. Sedangkan aplikasi basada berbasis website adalah aplikasi yang digunakan oleh admin bank sampah untuk mengelola data sampah, penjadwalan sampah dan penjualan sampah ke pelapak. Berdasarkan hasil deteksi dengan menggunakan SonarQube, memberikan hasil pengukuran untuk aplikasi basada berbasis mobile dengan hasil quality gate pada keseluruhan kode aplikasi dinyatakan Failed berwarna merah. Hal ini disebabkan karena terdapat 1 bug dengan nilai C, yang artinya terdapat 1 bug dengan indikasi major bug. Kemudian pengukuran terhadap aspek vurnerability dengan hasil 0 dan nilai A, yang artinya tidak ada issue dari kode program yang mengarah kepada kerentanan koding dari serangan keamanannya. Juga ditemukan 126 Code smells dengan estimasi waktu perbaikan 2 jam 34 menit. Dan terdeteksi duplicate code sebesar 10,7%. Hasil pendeteksian dari aplikasi basada berbasis mobile dapat dilihat pada gambar 3 berikut:

Gambar 3. Laporan analisis project basada berbasis mobile

Deteksi code smell pada aplikasi basada berbasis mobile masuk dalam kategori minor dan mayor. Kategori minor adalah dalam hal aturan untuk memberi nama variabel, parameter, definisi top-level, anggota kelas, dan konstruktor dengan huruf besar pada setiap kata, kecuali kata pertama, dan tanpa pemisah atau disebut dengan istilah LowerCamelCaseIdentifier. Berikut adalah tampilan hasil code smells dengan kasus LowerCamelCaseIdentifier pada gambar 4 berikut:

```
abstract class Routes {
  Routes._();
  static const HOME = _Paths.HOME;
  static const LOGIN = _Paths.LOGIN;
  static const REGISTER = _Paths.REGISTER;
  static const BASE_URL = "https://banksampah.pekanbaru.go.id";
  static const TOKEN = "token";
  static const USER_ID = "user_id";
  static const ROLE = "role";
  static const REGISTER_TOKEN = "register_token";
  static const REGISTER_NEXT = _Paths.REGISTER_NEXT;
  static const JENIS_SAMPAH = _Paths.JENIS_SAMPAH;
  static const KATEGORI_SAMPAH = _Paths.KATEGORI_SAMPAH;
  static const JUAL_SAMPAH = _Paths.JUAL_SAMPAH;
  static const FORM_JUAL = _Paths.FORM_JUAL;
  static const SEARCHMAP = _Paths.SEARCHMAP;
  static const PROFILE = _Paths.PROFILE;
  static const HISTORY = _Paths.HISTORY;
  static const EDIT_PROFILE = _Paths.EDIT_PROFILE;
}
```

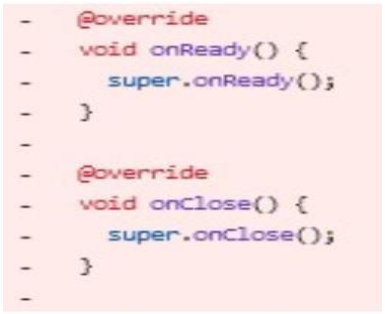
Gambar 4. Code Smell LowerCamelCaseIdentifier

Sebelumnya penamaan variabel ditulis dalam huruf Uppercase, dan disarankan untuk dirubah menjadi huruf LowerCase. Berikut adalah hasil perubahan kode untuk menyelesaikan code smells dengan kasus LowerCamelCaseIdentifier pada gambar 5 berikut:

```
abstract class Routes {
  Routes._();
  static const routeHome = _Paths.routeHome;
  static const routeLogin = _Paths.routeLogin;
  static const routeRegister = _Paths.routeRegister;
  static const baseUrl = "https://banksampah.pekanbaru.go.id";
  static const token = "token";
  static const userId = "user_id";
  static const role = "role";
  static const registerToken = "register_token";
  static const routeRegisterNext = _Paths.routeRegisterNext;
  static const routeJenisSampah = _Paths.routeJenisSampah;
  static const routeKategoriSampah = _Paths.routeKategoriSampah
  static const routeJualSampah = _Paths.routeJualSampah;
  static const routeFormJual = _Paths.routeFormJual;
  static const routeSearchMap = _Paths.routeSearchMap;
  static const routeProfile = _Paths.routeProfile;
  static const routeHistory = _Paths.routeHistory;
  static const routeEditProfile = _Paths.routeEditProfile;
}
```

Gambar 5. Perbaikan Code Program untuk kasus Code Smells LowerCaseCamelIdentifier

Selanjutnya deteksi code smells untuk kategori mayor disebabkan oleh “Unnecessary Override”. Unnecessary override" adalah istilah yang digunakan dalam pemrograman untuk merujuk pada situasi di mana sebuah metode atau fungsi dari kelas dasar (base class) atau kelas antarmuka (interface class) yang mendefinisikan kode yang sama persis di dalam kelas turunan (derived class). Hal ini dianggap sebagai kode program yang tidak perlu dan dapat menambah kerumitan kode tanpa memberikan manfaat yang signifikan. Berikut adalah tampilan deteksi code smells untuk kasus Unnecessary Override pada gambar 6. Sehingga code program disarankan untuk dihapus.



Gambar 6. Deteksi Code Smells untuk Kasus Unnecessary Override

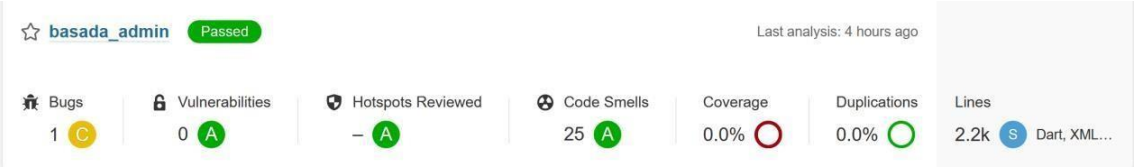
Berikut adalah hasil rangkuman identifikasi code smells pada aplikasi basada berbasis mobile terdapat 126 code smells yang dapat dilihat pada tabel 1 berikut:

Tabel 1. Hasil identifikasi code smells pada aplikasi basada berbasis mobile

Jenis Code Smell	Tindakan	Jumlah
(Dart) Constant identifier names	PREFER using lowerCamelCase for constant names.	15
(Dart) Unnecessary new	AVOID new keyword to create instances.	3
(Dart) Unnecessary this	DON'T use this when not needed to avoid shadowing.	3
(Dart) Prefer collection literals	DO use collection literals when possible.	2
(Dart) No leading underscores for local identifiers	DON'T use a leading underscore for identifiers that aren't private.	1
(Dart) Non constant identifier names	DO name non-constant identifiers using lowerCamelCase.	1
Total		126

Hasil pengukuran quality gate secara keseluruhan pada project aplikasi basada berbasis website adalah Passed berwarna hijau. Tidak terdapat issue vurnerablities sehingga nilainya 0 dengan status nilai A. Kemudian terdeteksi 25 code smells dengan nilai A, yang artinya adalah persentase jumlah code smells <= 5% dari keseluruhan jumlah line of code program. Dan tidak terdapat kode

program yang duplicated. Berikut tampihan hasil pengukuran project basada berbasis website pada gambar 7 berikut:



Gambar 7. Hasil pengukuran quality gate pada project basada berbasis website

Deteksi 25 code smells pada project basada berbasis website masuk dalam kategori minor. Berikut adalah hasil identifikasi code smells yang dapat dilihat pada tabel 2 berikut:

Tabel 2. Hasil deteksi code smells pada aplikasi basada berbasis website

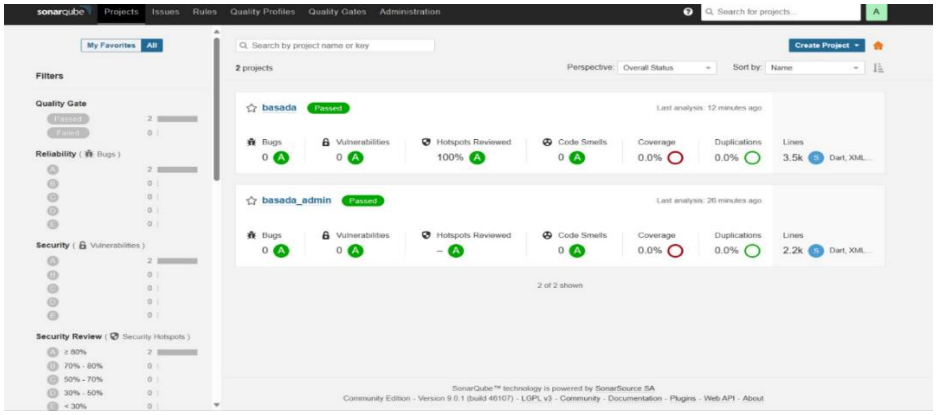
Jenis Code Smell	Tindakan	Jumlah
(Dart) Unnecessary this	DON'T use this when not needed to avoid shadowing.	51
(Dart) Constant identifier names	PREFER using lowerCamelCase for constant names.	30
(Dart) Unnecessary new	AVOID new keyword to create instances.	15
(Dart) Prefer collection literals	DO use collection literals when possible.	12
(Dart) Non constant identifier names	DO name non-constant identifiers using lowerCamelCase.	8
(Dart) Unnecessary overrides	DON'T override a method to do a super method invocation with same parameters.	7
(Dart) Always declare return types	DO declare method return types.	2
(Dart) Prefer interpolation to compose strings	PREFER using interpolation to compose strings and values.	1
(HTML) "<html>" elementshould have a language attribute	The <html> element should provide the lang and/or xml:lang attribute inorder to identify the default language of a document.	1
Total		25

Berikut adalah hasil pengukuran dari 2 project Bank Sampah DLHK Kota Pekanbaru yang dapat dilihat pada tabel 3 berikut:

Tabel 3. Hasil pengukuran dari dua project Bank Sampah DLHK Kota Pekanbaru

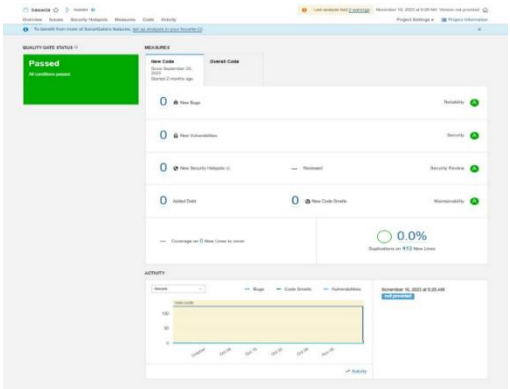
Aplikasi	Bugs	Vulnerability	Security	Code Smell	Duplication
Basada	1	0	2	126	10.7%
Basada Admin	1	0	0	25	0%

Setelah perbaikan dilakukan, maka pengukuran terhadap kualitas kode program kembali dilakukan. Adapun hasil pengukuran setelah melakukan perbaikan dapat dilihat pada tampilan quality gate keseluruhan pada SonarQube pada gambar 8 berikut:



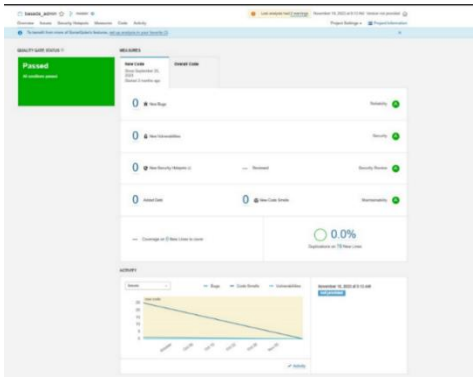
Gambar 8. Hasil perbaikan code smells pada halaman dashboard SonarQube

Pengukuran kualitas kode program pada dua aplikasi Bank sampah DLHK Kota Pekanbaru dilakukan kembali setelah memperbaiki bug dan code smells yang terdeteksi serta memperbaiki duplicate code program. Sehingga pada project aplikasi Basada yang pada pengukuran sebelumnya berstatus Failed (berwarna merah) setelah diperbaiki berubah status menjadi Passed (berwarna hijau). Adapun detil dari hasil pengukuran setelah perbaikan pada project Basada berbasis mobile dapat dilihat pada gambar 9 berikut:



Gambar 9. Hasil pengukuran project basada berbasis mobile setelah perbaikan

Sedangkan hasil pengukuran kembali pada project aplikasi Basada berbasis website dapat dilihat pada gambar 10 berikut:



Gambar 10. Hasil pengukuran project aplikasi basada berbasis website setelah perbaikan

Dengan melakukan perbaikan terhadap kualitas kode program maka program pada dua project aplikasi Bank Sampah DLHK Kota Pekanbaru sudah dinyatakan secara kualitas memenuhi untuk dapat dimodifikasi dan dimaintain terhadap penambahan fitur baru.

5. KESIMPULAN

Berdasarkan pada penelitian ini dapat disimpulkan bahwa penggunaan tools SonarQube untuk menguji kualitas kode program, khususnya untuk mendeteksi adanya bugs, code smells dan duplicate code dapat dijadikan sebagai acuan untuk standarisasi peningkatan kualitas kode program. Penggunaan SonarQube pada dua project aplikasi Bank sampah DLHK kota Pekanbaru, menghasilkan pengukuran pada project aplikasi basada berbasis mobile terdapat 1 bug dengan kategori major, 126 code smells dan 10,7% duplicated kode program. Sedangkan pada project aplikasi basada berbasis website ditemukan ada 1 bug dengan kategori minor, 25 code smells dan 0% duplicated kode program. Setelah dilakukan perbaikan dari hasil temuan, maka dilakukan pengujian ulang menggunakan tool yang sama dan diperoleh hasil bahwa tidak ada terdeteksi code smell pada aplikasi mobile maupun aplikasi web Basada.

6. DAFTAR PUSTAKA

[1] Rinci Kembang Hapsari and M Jauhari Husen, “Estimasi Kualitas Perangkat Lunak Berdasarkan Pengukuran Kompleksitas Menggunakan Metrik Function Oriented,” *Semin. Nas. Sains dan Teknol. Terap. III*, 2015.

[2] S. F. Sujadi, “Evaluasi Deteksi Smell Code dan Anti Pattern pada Aplikasi Berbasis Java,” *J. Tek. Inform. dan Sist. Inf.*, vol. 5, pp. 370–385, 2019, doi: https://doi.org/10.28932/jutisi.v5i3.1981.

[3] D. Albuquerque, E. Guimarães, M. Perkusich, H. Almeida, and A. Perkusich, “Integrating Interactive Detection of Code Smells into Scrum: Feasibility, Benefits, and Challenges,” *Appl. Sci.*, vol. 13, no. 15, Aug. 2023, doi: 10.3390/app13158770.

[4] A. Löwe, S. Bampovits, and F. Palma, “Do Software Code Smell Checkers Smell Themselves? A Self Reflection,” 2020.

[5] M. F. Firdaus, B. Priyambadha, and F. Pradana, “Pembangunan Sistem Untuk Pendeteksian Code Smells Refused Bequest,” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 12, pp. 6722–6728, 2018.

[6] V. Virdus, B. Priyambadha, and A. A. Soebroto, “Pembangunan Sistem Aplikasi Deteksi Code Smell berdasarkan Metrik Feature Envy,” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 8, pp. 7500–7506, 2019.

[7] Anil P. Mathew and Filipe A. R. Capela, “An Analysis on Code Smell Detection Tools,” *17th SC@RUG 2020 Proc. 2019-2020*, 2020.

[8] A. S. Cairo, G. de F. Carneiro, and M. P. Monteiro, "The impact of code smells on software bugs: A systematic literature review," *Information (Switzerland)*, vol. 9, no. 11. MDPI AG, Nov. 2018. doi: 10.3390/info9110273.
[9] X. Liu and C. Zhang, "The detection of code smell on software development: a mapping study," in *5th International Conference on Machienary, Materials and Computing Technology*, 2017.
[10] Muhammad Syarif, "Mengenal SonarQube Alat Untuk Cek Kualitas Program Secara Otomatis."
[11] Yusfia Hafid Aristyagama, "Framework Deteksi Bad Smell Code Semi Otomatis untuk Pemrograman Tim," 2016.
[12] H. P. Putro and I. Liem, "Deteksi Code Smell Pada Kode Program Dalam Representasi AST dengan Pendekatan By Rules," in *Seminar Nasional Aplikasi Teknologi Informasi*, 2010, pp. 1907–5022.
[13] M. R. Manero, "Measuring The Impact of SonarQube on The Development Velocity Using Regresion Analysis," 2023.
[14] A. Febriana Rahmawati and Y. Alfa Susetyo, "Analisis Quality Code Menggunakan Sonarqube Dalam Suatu Aplikasi Berbasis Laravel," *J. Penerapan Teknol. Inf. dan Komun.*, vol. 2, no. 2, pp. 89–103, 2023.
[15] Sonar, "What is a code smell?," https://www.sonarsource.com/learn/code-smells/.
[16] J. Larson, "What Are Code Smells? (Examples With Solutions)," https://builtin.com/software-engineering-perspectives/code-smells.
[17] L. Dohm, "Code Smells," https://github.com/lee-dohm/code-smells.
[18] A. Kadangode, "Exploring Code Smells: Identifying and Solving Common Issues," https://dev.to/akhilani/exploring-code-smells-identifying-and-solving-common-issues-4bo0.