*TRANSMISI: JURNAL ILMIAH TEKNIK ELEKTRO, 26, (3), JULI 2024*
*p-ISSN 1411-0814 e-ISSN 2407-6422*

# IMPLEMENTASI DEVOPS PADA PENGEMBANGAN APLIKASI ANDROID PENDETEKSI KUALITAS BERAS BERBASIS MACHINE LEARNING

Febriyanti Paramudita, Mulki Indana Zulfa[*)] dan Acep Taryana

Jurusan Teknik Elektro, Universitas Jenderal Soedirman, Purbalingga, Indonesia

[*)]*Penulis korespondensi, E-mail: mulki_indanazulfa@unsoed.ac.id*

## Abstrak

Beras merupakan makanan pokok sebagian besar penduduk Indonesia. Permintaannya cenderung meningkat sekitar 1% setiap tahun hingga tahun 2050. Permintaan yang terus meningkat tersebut menuntut standar kualitas yang tinggi. Saat ini, Badan Urusan Logistik (BULOG) masih menggunakan metode konvensional dalam pemantauan kualitas beras yang rentan terhadap *human error* dan subjektivitas. Masalah yang muncul dari metode ini adalah ketidakakuratan dalam penilaian kualitas beras yang berdampak pada distribusi dan kepercayaan konsumen. Dalam penelitian ini, penulis memanfaatkan teknologi informasi, khususnya aplikasi Android dengan pendekatan DevOps, untuk meningkatkan efisiensi dan objektivitas dalam mendeteksi kualitas beras. Hasil penelitian menunjukkan bahwa implementasi DevOps membawa dampak positif terhadap kualitas dan stabilitas aplikasi, dengan 60 *pipeline* yang membentuk alur kerja yang fleksibel. Melalui pemantauan, hasil menunjukkan variasi waktu mulai aplikasi dari 338ms hingga 1.61s, respons perangkat fisik adalah 3.97s, tidak ada kasus *slow rendering*, kualitas kode mencapai 100%, dan ukuran *payload* berkisar antara 3.41KB dan 6.42KB, sesuai dengan kompleksitas jumlah data yang diperlukan oleh masing-masing fitur. Dengan demikian, aplikasi ini berpotensi memberikan solusi efektif untuk memonitor dan meningkatkan kualitas beras di pasar, mengurangi ketergantungan pada metode konvensional yang kurang efisien.

*Kata kunci: DevOps, aplikasi, android, machine learning, kualitas beras*

## *Abstract*

*Rice serves as the staple food for a significant portion of the Indonesian population. Its demand tends to increase by approximately 1% annually until the year 2050. This continuous rise in demand necessitates stringent quality standards. Presently, the Logistics Agency (BULOG) relies on conventional methods to monitor rice quality, which are susceptible to human error and subjectivity. The issue arising from these methods lies in the inaccuracies of rice quality assessments, impacting both distribution and consumer trust. In this study, the author leverages information technology, particularly Android applications employing the DevOps approach, to enhance the efficiency and objectivity of rice quality detection. The research findings indicate that the implementation of DevOps positively affects the quality and stability of the application, with 60 pipelines forming a flexible workflow. Through monitoring, results reveal a variation in application startup times ranging from 338ms to 1.61s, with a physical device response time of 3.97s. No instances of slow rendering were observed, and the code quality reached 100%, with payload sizes ranging from 3.41KB to 6.42KB, aligned with the complexity of data required by each feature. Thus, this application holds potential to offer an effective solution for monitoring and enhancing rice quality in the market, thereby reducing reliance on less efficient conventional methods.*

*Keywords: DevOps, application, android, machine learning, rice quality*

## 1. Pendahuluan

Beras merupakan makanan pokok dan bahan utama dalam berbagai produk pangan lainnya [1]. Permintaan beras akan cenderung meningkat sejalan dengan meningkatnya jumlah penduduk. Permintaan beras di seluruh Indonesia terus meningkat sekitar 1,00 persen di setiap tahun hingga tahun 2050 [2].

Mutu beras dipengaruhi oleh beberapa ciri fisik seperti (1) ukuran dan bentuk butiran, (2) tingkat derajat sosoh, (3) kejernihan, (4) kebersihan dan kemurniannya [3]. Kualitas beras dapat dikategorikan berdasarkan bentuk dan warna. Semakin putih, bersih, dan utuh beras, semakin baik kualitasnya. Berdasarkan kualitasnya, beras dapat dikategorikan ke dalam kualitas beras premium, medium, dan tidak layak [4].

Masyarakat perlu cermat dalam memeriksa kualitas beras, apakah beras tersebut cocok untuk dimasak atau tidak. Namun, kenyataannya banyak masyarakat yang belum memiliki pengetahuan khusus tentang cara mengenali kualitas beras yang sesuai untuk dimasak [5]. Badan Urusan Logistik (BULOG) bertanggung jawab atas

logistik dan pangan [6]. Pemantauan kualitas beras di BULOG masih menggunakan metode konvensional, seperti mengukur 10% dari total beras sebagai sampel uji, memisahkan butir beras, dan menimbangnya [7]. Metode manual ini rentan terhadap *human error* dan subjektivitas. Meskipun BULOG memiliki alat bernama Milling Level Meter untuk mendeteksi kualitas beras, harga yang mahal membuat banyak petani dan masyarakat memilih cara manual untuk mengetahui kualitas beras menggunakan indra perasa mereka [8].

Pemanfaatan teknologi informasi, khususnya melalui aplikasi Android, dapat efektif mendeteksi kualitas beras [9]. Aplikasi ini menyediakan hasil konsisten dan objektif, mengurangi subjektivitas dalam penilaian manusia. Dengan teknologi *machine learning*, aplikasi dapat mengidentifikasi karakteristik beras secara akurat, memungkinkan pemantauan berkelanjutan dalam waktu nyata, dan memfasilitasi pengambilan keputusan cepat.

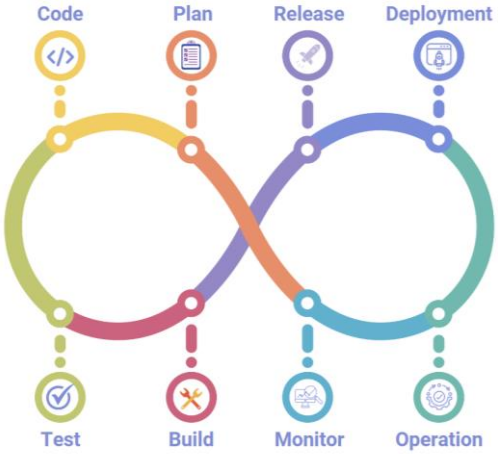
Berbeda dengan penelitian Mustofah dan Utami (2019) [7] yang menggunakan perangkat keras Arduino Uno dan fokus pada kadar air dan berat butir menir, penilitian ini mengadopsi pendekatan pengembangan perangkat lunak dengan DevOps dan machine learning.

Dharmik et al. (2022) [10] menggunakan RCNN untuk analisis kualitas beras berbasis pengolahan citra, namun penelitian ini berfokus pada aplikasi bergerak yang memanfaatkan CNN untuk klasifikasi berdasarkan warna dan tekstur beras. Sementara itu, Ardi (2021) [11] mengembangkan aplikasi serupa berbasis Android menggunakan metode k-Nearest Neighbor, penelitian ini menggunakan algoritma CNN untuk meningkatkan akurasi deteksi.

Inovasi utama dalam penelitian ini terletak pada penggunaan teknologi machine learning dengan CNN dan penerapan metodologi DevOps untuk mengoptimalkan siklus pengembangan aplikasi android. DevOps memungkinkan kolaborasi antara tim pengembang dan tim operasi, memastikan pengembangan dan pengelolaan perangkat lunak yang cepat, andal, dan berkualitas tinggi [12][13]. Implementasi DevOps dalam pengembangan aplikasi android memiliki manfaat signifikan, memecahkan masalah kritis yang dapat menghambat pengembangan software [14][15].

## 2. Metode

Metode pengembangan akan menerapkan pendekatan DevOps dengan mengikuti serangkaian Langkah. Alur tahapan pengembangan dapat dilihat pada Gambar 1.




**Gambar 1. Alur Tahap Pengembangan**

**1. Tahap Perencanaan (*Plan*)**

Pada tahap perencanaan, tim Android merencanakan fungsionalitas aplikasi Android seperti desain UI/UX, *library* yang akan digunakan. Sedangkan tim *machine learning* merencanakan model ML yang akan dibangun dimulai dari mencari *dataset* yang akan digunakan. Selain itu kedua tim ini bekerja sama dalam menentukan *timeline* dan manajemen tugas [15].

**2. Tahap Pengembangan Kode (*Code*)**

Tim android mulai menulis kode aplikasi, termasuk antarmuka pengguna dan logika aplikasi. Sedangkan tim *machine learning* akan memilih algoritma dan arsitektur model yang akan digunakan, kemudian menulis kode yang mencakup definisi model *machine learning*, dan penyesuaian hyperparameter mode, seperti jumlah *epoch* [15].

**3. Tahap Pengujian (*Test*)**

Tahap pengujian terdiri dari dua fase. Fase pertama melibatkan pemeriksaan keseluruhan proyek dengan melakukan scanning code. Fase kedua dilakukan melalui uji unit untuk memastikan keberhasilan fungsi setiap komponen secara terisolasi [15].

**4. Tahap Pembangunan (*Build*)**

Tahap pembangunan melibatkan proses konversi kode sumber aplikasi Android menjadi paket aplikasi yang dapat diinstal di perangkat, menggunakan alat pembangunan Gradle [16].

**5. Tahap Perilisan (*Release*)**

Setelah berhasil melewati tahap pengujian, tim Android siap untuk merilis aplikasi ke GitHub. Dalam konteks ini, tahap rilis tidak berarti aplikasi siap untuk dipublikasikan kepada pengguna akhir. Pada tahap ini, lebih tepatnya merupakan tahap "cek poin" [15].

**6. Tahap Peluncuran (*Deployment*)**

Tahap ini akan dibagi menjadi dua paradigma, paradigma pertama kedua adalah men-*deploy* atau mengintegrasikan model ML ke dalam aplikasi android. Sedangkan paradigma kedua melibatkan instalasi dan konfigurasi aplikasi versi final di lingkungan produksi atau di perangkat pengguna [15].

**7. Tahap Operasi (*Operation*)**

Pada tahap ini, pengoperasian aplikasi dilakukan dengan tujuan menguji aplikasi versi final sebagaimana pengguna sebenarnya melakukannya. Tindakan ini dimaksudkan untuk mengevaluasi langsung pengalaman pengguna (*user experience*) dan memastikan bahwa aplikasi berkinerja sesuai dengan harapan pengguna [16].

**8. Tahap Pemantauan (*Monitor*)**

Pada tahap ini, dilakukan pemantauan terhadap aplikasi final yang telah di-*deploy*. Dalam penelitian ini, pemantauan kinerja aplikasi dilakukan melalui dasbor performa yang terdapat di Firebase Console [16].

## 3. Hasil dan Pembahasan

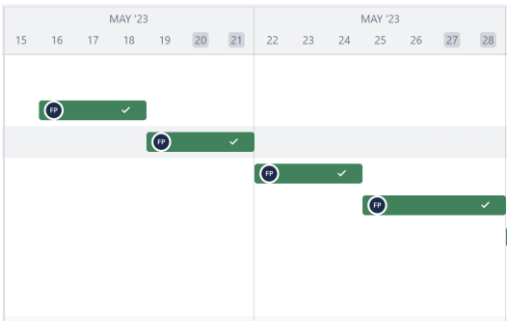
### 3.1. Perencanaan (*Plan*)

Tahap perencanaan memuat dua jenis pekerjaan yang berbeda yaitu perencanaan kolaboratif dan individu.

**1. Perencanaan Kolaboratif**

A. Timeline

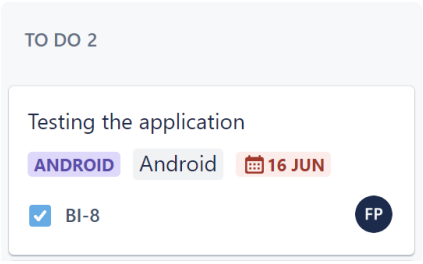
*Timeline* untuk proyek ini disusun menggunakan perangkat lunak Jira Atlassian. Proyek ini memiliki durasi lima minggu, dimulai dari tanggal 16 Mei 2023 hingga 16 Juni 2023. Setiap tim, baik tim Android maupun tim ML memiliki tanggung jawab dan pekerjaan tertentu yang harus diselesaikan sesuai dengan jadwal yang telah ditetapkan. Gambar 2 merupakan *timeline* dari proyek ini. *Timeline* ini terbagi menjadi kategori *TODO* (elum dikerjakan), *IN PROGRESS* (sedang dikerjakan), dan *DONE* (Sudah dikerjakan).




**Gambar 2. Timeline Proyek**

B. Board

Penerapan Board memungkinkan kedua tim untuk secara terorganisir memahami alur kerja dan melihat kemajuan proyek, sehingga kolaborasi antara kedua tim menjadi lebih efisien. Gambar 3 merupakan salah satu *board TODO* dengan kode *card* BI-8 menjelaskan mengenai tugas yang harus dikerjakan oleh tim yang bertanggung jawab (Tim Android), dan tanggal tenggat waktu (16 Juni).




**Gambar 3. Board Proyek**

C. Analisis Kebutuhan

Kebutuhan pengguna akan dibagi menjadi dua yaitu kebutuhan fungsional dan non-fungsional. Masing-masing kebutuhan dijelaskan pada Tabel 1 dan Tabel 2.

**Tabel 1. Kebutuhan Non-Fungsional**

| No. | Kebutuhan | Deskripsi |
|---|---|---|
| 1. | Aplikasi android | Menyediakan sistem pendeteksi kualitas beras berbasis android agar memudahkan pengambilan gambar. |
| 2. | Peforma | Aplikasi harus memberikan hasil deteksi kualitas beras dalam waktu kurang dari 3 detik setelah pengguna menekan tombol "Deteksi" |
| 3. | Kemudahan Penggunaan | Antarmuka pengguna harus dirancang sedemikan rupa agar mudah digunakan oleh pengguna. |
| 4. | Interoperabilitas | Aplikasi harus dapat berintegrasi dengan kamera ponsel dan mendukung berbagai model perangkat android. |

**Tabel 2. Kebutuhan Fungsional**

| No. | Kebutuhan | Deskripsi |
|---|---|---|
| 1. | Aplikasi mampu melakukan deteksi melalui kamera | Tujuan: Aplikasi dapat mendeteksi beras secara langsung melalui kamera.<br>Input: Objek beras yang diarahkan ke kamera.<br>Operasi: Menampilkan hasil deteksi beras berdasarkan akurasi tertinggi.<br>Output: User memperoleh hasil deteksi kualitas beras. |
| 2. | Aplikasi mampu melakukan deteksi melalui galeri. | Tujuan: Aplikasi dapat mendeteksi beras melalui gambar yang diunggah dari galeri.<br>Input: Gambar beras<br>Operasi: Menampilkan hasil deteksi beras berdasarkan akurasi tertinggi.<br>Output: User memperoleh hasil deteksi kualitas beras. |

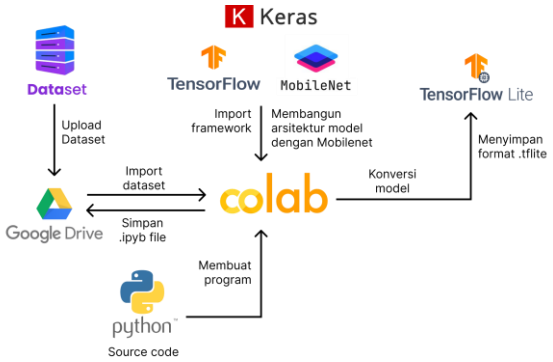
**2. Perencanaan Individu**

A. Tim Android

Pada tahap ini, tim Android akan merencanakan antarmuka aplikasi dengan merancang desain UI/UX. Fokusnya adalah pada aspek visual dan fungsional aplikasi. Total desain halaman yang akan dirancang adalah 10 halaman. Desain UI aplikasi dapat diakses di Figma proyek ini.

B. Tim Machine Learning

Pada tahap ini, menentukan langkah-langkah yang diperlukan dalam pembuatan model. Gambar 4 merupakan alur yang memvisualisasikan rencana dari tim ML untuk pembuatan model.



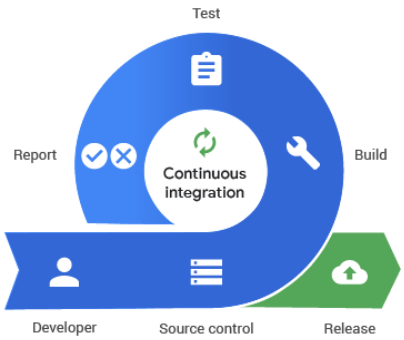

**Gambar 4. Alur Pengembangan Model**

Pertama, dataset diunggah ke Google Drive, lalu diimpor ke Google Colab untuk proses pemrosesan dan pembangunan model. Setelah model dibuat dan diuji di Google Colab, hasilnya disimpan kembali ke Google Drive. Selanjutnya, pengguna menulis kode Python di Google Colab, menggunakan framework TensorFlow, dan membangun model Machine Learning dengan arsitektur MobileNet. Model yang sudah dibuat diubah ke format .tflite untuk efisiensi saat diimplementasikan pada aplikasi Android. Proses ini memastikan model dapat beroperasi secara optimal pada perangkat seluler.

**3.2. Kode (*Code*)**

**1. Tim Android**

Tim Android menggunakan Android Studio untuk mengembangkan aplikasi Android, sementara GitHub digunakan sebagai platform kontrol versi. Setiap kali kode di-push ke repositori GitHub, CircleCI secara otomatis memulai dan mengeksekusi workflow build, test, dan release. Otomatisasi ini meningkatkan efisiensi dan konsistensi dalam alur kerja pengembangan serta memastikan pengujian otomatis setiap perubahan kode [17]. Gambar 5 menunjukkan alur *continuous integration*.




**Gambar 5. Alur Continous Integration**

A. GitHub

Tim Android menginisiasi repositori baru di GitHub sebagai penyimpanan untuk kode sumber aplikasi. Tim Android secara teratur melakukan *push* kode ke repositori setiap kali ada perubahan atau penambahan kode. GitHub menyediakan riwayat perubahan (*commit history*) yang memungkinkan tim untuk melacak evolusi kode dan mengembalikan ke versi sebelumnya jika perlu, serta membantu memastikan transparansi dan kelola riwayat kode. Kode sumber lengkap aplikasi dapat diakses di Github Repository.

B. Android Studio

Proyek aplikasi ini memiliki kelas MainActivity.kt, SplashScreen.kt, dan 3 folder utama yaitu folder Data, Detection, dan UI. Folder Data terdiri dari 2 sub folder yaitu Model dan Retrofit. Folder model berisi kelas yang digunakan untuk mengelola respons terkait fitur-fitur yang memerlukan API eksternal. Folder Retrofit berisi kelas-kelas untuk mengonfigurasi API dalam. Folder Detection berisi kelas-kelas untuk mengonfigurasi fitur deteksi baik melalui kamera maupun galeri. Sedangkan folder UI merupakan kode untuk menampilkan antarmuka setiap lama ke pengguna.

C. CircleCI

CircleCI adalah platform Continous Integration yang mengotomatisasi langkah-langkah build, test, dan release setiap kali tim Android melakukan push ke repositori GitHub [18]. File konfigurasi 'config.yaml' di dalam folder khusus '.circleci' mendefinisikan proses tersebut dalam format YAML.

**2. Tim Machine Learning**

Tim ML menggunakan Google Colab sebagai platform utama untuk kode ML, dengan TensorFlow dan Keras sebagai framework utama. Mereka memilih arsitektur MobileNet untuk model ML, terkenal karena efisiensinya dalam tugas computer vision pada perangkat seluler. Langkah-langkah pengkodean termasuk impor dataset dari Google Drive, inspeksi dataset, augmentasi data dengan ImageDataGenerator, dan pengembangan arsitektur model menggunakan TensorFlow dan Keras.

### 3.3. Pengujian (*Test*)

**1. *Scanning Code***

Pada tahap awal pengujian, kode yang dikirim oleh tim Android ke repositori GitHub menjalani *scanning code* untuk mengidentifikasi potensi eror atau peringatan dengan menggunakan alat bantu linters. Proses ini diatur dalam berkas config.yml dengan perintah `./gradlew lint` test. Tim Android menghadapi dua kegagalan dalam *scanning code*, terjadi pada pipeline 1 dan 39. Kegagalan pada *pipeline* 1 terkait dengan masalah kompatibilitas dengan API Android, dimana beberapa dependensi memerlukan versi kompilasi minimum Android-34.

Kegagalan pada pipeline 39 terjadi karena sumber daya 'ic_search' tidak ditemukan di folder 'drawable'. Tim Android harus memastikan sumber daya telah ditambahkan ke folder yang sesuai.

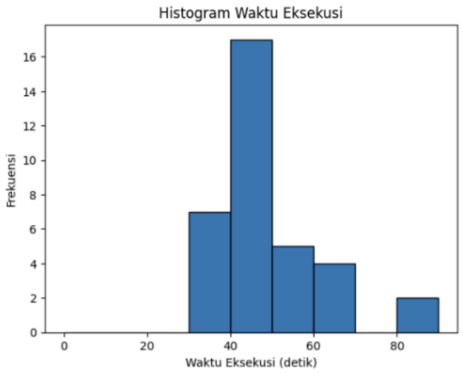
**2. *Unit Tests***

Setelah *scanning code*, CircleCI melanjutkan ke *unit tests*, yaitu pengujian pada setiap fungsi dan metode secara terpisah. Proyek ini melakukan 5 *unit test* untuk menguji logika implementasi dalam kode.

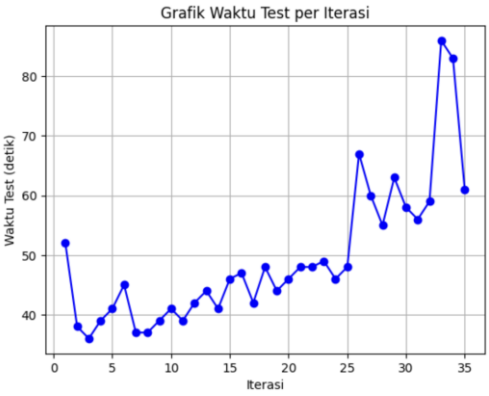
A. Analisis *Time-based metrics*

Metrik ini mengevaluasi efisiensi waktu yang dibutuhkan oleh proses Continous Integration (CI) dalam menyelesaikan proses pengujian [13]. Gambar 6 merupakan histogram waktu eksekusi CircleCI dalam proses pengujian.

Rentang waktu eksekusi dibagi ke dalam interval 10 detik. Histogram menunjukkan frekuensi iterasi dengan waktu eksekusi tertentu. Mayoritas iterasi berlangsung antara 30 hingga 50 detik, dengan puncak frekuensi di sekitar 40-50 detik. Namun, beberapa iterasi memerlukan waktu lebih lama, menandakan adanya perubahan kompleksitas baru dalam kode. Gambar 7 menunjukkan performa dan konsistensi proses tes selama 35 iterasi. Peningkatan waktu tes terjadi karena penambahan jumlah *unit test* pada beberapa iterasi, yang memperpanjang waktu total yang dibutuhkan.




**Gambar 6. Histogram Waktu Eksekusi Test**




**Gambar 7. Grafik Waktu Test per Iterasi**

B. Analisis *Quality-based metrics*

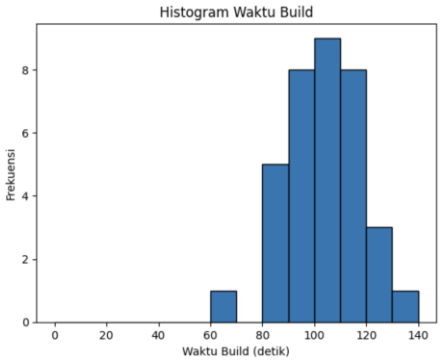
Analisis ini mengevaluasi kesuksesan dan kegagalan tes dalam 60 *pipeline*. Dari 60 *pipeline* yang dievaluasi, 35 memiliki *success rate* 100%, menunjukkan tes berhasil tanpa kegagalan. Namun, pada *pipeline* 23, hanya 3 dari 6 tes berhasil, dengan *success rate* 50%. Sedangkan pada *pipeline* 24, *success rate* mencapai 67%.

### 3.4. Pembangunan (*Build*)

Setelah pengujian, CircleCI melakukan proses *build* secara otomatis menggunakan *command* `./gradlew build` dari konfigurasi (config.yml). Proses *build* memanfaatkan *platform* Docker untuk mengatur konfigurasi dengan lebih baik dan mengisolasi dependensi proyek dalam *container*, mencegah konflik antar lingkungan pengembangan [19].

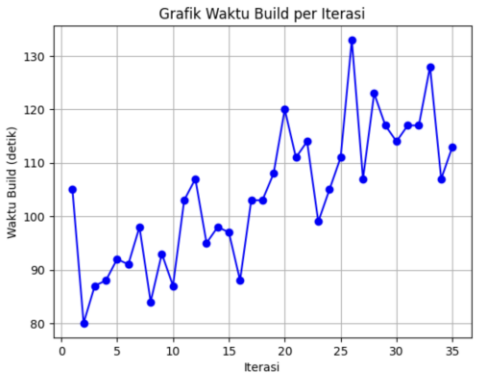
### 1. Analisis *Time-based metrics*

Metrik waktu dari proses *build* memiliki dampak besar pada produktivitas dan kualitas perangkat lunak. Gambar 8 menunjukkan histogram waktu eksekusi proses *build*.




**Gambar 8. Histogram Waktu Build**

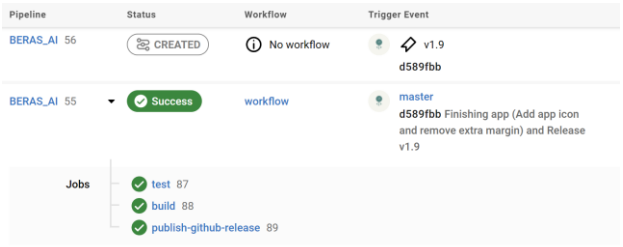
Histogram menunjukkan rata-rata waktu *build* sekitar 100 detik, dengan distribusi cenderung normal dalam rentang 80-110 detik. Waktu build tercepat adalah 60-70 detik, sementara yang terlama mencapai 130 detik. Frekuensi tertinggi terjadi pada 90-100 detik, menandakan sebagian besar iterasi membutuhkan waktu dalam rentang tersebut. Gambar 9 menunjukkan peforma dan konsistensi proses *build* selama 35 iterasi. Terjadi peningkatan waktu *build* pada 10 iterasi terakhir akibat penambahan sumber daya seperti model *machine learning* dan *dependencies* tambahan, yang dapat meningkatkan kompleksitas dan memperlambat proses *build*.




**Gambar 9. Grafik Waktu Build per Iterasi**

### 3.5. Perilisan (*Release*)

Proses perilisan otomatis memanfaatkan berkas konfigurasi untuk menentukan versi, membuat permintaan POST ke API GitHub, melakukan autentikasi dengan token GitHub, mengisi informasi rilis, dan memberikan keterangan tambahan tentang pembaruan. Gambar 10 menunjukkan pipeline pada *CircleCI* yang berhasil menyelesaikan seluruh tahap *workflow*, termasuk *test*, *build*, dan *release*, hanya dengan satu *push code*. Status "*CREATED*" menandakan bahwa proses telah sukses dan rilis aplikasi telah berhasil dibuat.




**Gambar 10. Daftar Rilis di GitHub**

Untuk representasi dari tahap perilisan dijelaskan pada Tabel 3.

**Tabel 3. Tabel Perilisan**

| No | Versi Rilis | Deskripsi Rilis |
|---|---|---|
| 1. | 1.0 | • Menambahkan splash screen sebagai layar pertama yang muncul saat aplikasi dijalankan.<br>• Menambahkan menu Home yang berisi informasi tentang tengkulak beras dan fitur pencarian. |
| 2. | 1.1 | • Menambahkan halaman “Harga Beras” yang menampilkan informasi harga beras di setiap provinsi. |
| 3. | 1.2 | • Menambahkan halaman “Deskripsi Beras” yang memberikan informasi tentang 3 jenis kualitas beras. |
| 4. | 1.3 | • Menambahkan halaman Deteksi dengan opsi deteksi melalui kamera dan galeri. |
| 5. | 1.4 | • Menambahkan fitur deteksi melalui kamera dengan tampilan kameran dan tombol deteksi. |
| 6. | 1.5 | • Menambahkan fitur deteksi melalui galeri dengan opsi memilih gambar dari galeri dan tombol deteksi. |
| 7. | 1.6 | • Menambahkan halaman Profile dengan menu Bantuan dan Tentang Aplikasi. |
| 8. | 1.7 | • Menambahkan halaman Tentang Aplikasi yang memberikan informasi tentang aplikasi BERAS.AI dan pengembangannya, |
| 9. | 1.8 | • Menambahkan halaman Bantuan dengan informasi tentang cara kerja dan penggunaan aplikasi. |
| 10. | 1,9 | • Menambahkan ikon aplikasi dan menghilangkan margin ekstra pada fragment. |

### 3.6. Peluncuran (*Deployment*)

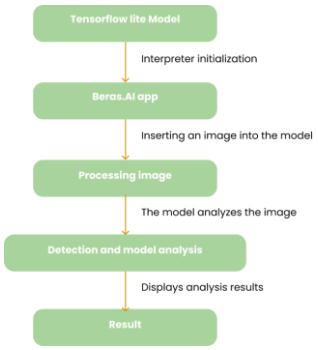
Dalam tahap peluncuran proyek ini, diterapkan dua paradigma utama: integrasi model machine learning ke aplikasi Android untuk meningkatkan fungsionalitas, dan distribusi aplikasi.

#### 1. *Deployment* Model Machine Learning

Proses deployment model ML ke dalam aplikasi android menggunakan format TensorFlow Lite (tflite) melibatkan

beberapa langkah kunci. Gambar 11 menjelaskan langkah-langkah model tensorflow lite dalam memproses citra pada aplikasi android.

Pertama, inisialisasi interpreter TensorFlow Lite model pada aplikasi. Selanjutnya, pengguna dapat memasukkan gambar ke dalam model untuk diproses. Model TensorFlow Lite kemudian menganalisis gambar tersebut, melakukan deteksi objek, dan menganalisis hasilnya. Hasil analisis kemudian ditampilkan pada aplikasi, memberikan pengguna informasi yang relevan berdasarkan deteksi objek yang dilakukan oleh model ML.




**Gambar 11. Proses Deteksi Objek dengan TensorFlow Lite Model pada Aplikasi**

## 2. *Deployment* Aplikasi ke Firebase

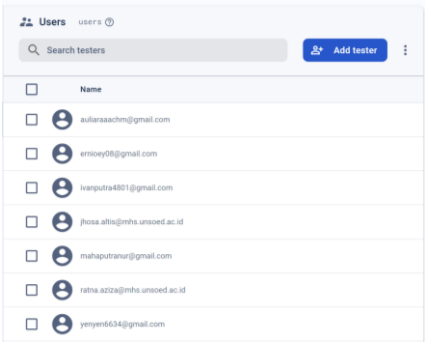
Proses *deployment* aplikasi menggunakan Firebase App Distribution, *platform* untuk distribusi efisien ke pengguna atau perangkat yang ditentukan. Setelah aplikasi diunggah ke Firebase Console, pengembang memilih grup pengguna atau perangkat. Firebase App Distribution secara otomatis mengirim undangan distribusi kepada pengguna atau perangkat terpilih, memudahkan mereka mengunduh dan menginstal versi terbaru aplikasi. Gambar 12 menunjukkan proses pendistribusian aplikasi kepada tim pengembang, melibatkan undangan, penerimaan undangan, dan pengunduhan aplikasi.


**Gambar 12. Distribusi Aplikasi ke Pengguna Terpilih**

Metode alternatif distribusi melibatkan membagikan link distribusi kepada berbagai pihak. Pengguna yang mengunjungi link tersebut diminta untuk memasukkan alamat email mereka, yang akan secara otomatis tercatat di console dan ditambahkan ke dalam grup "Users". Grup "Users" memungkinkan pengembang untuk mendistribusikan versi terbaru aplikasi kepada seluruh anggota grup. Gambar 13 menunjukkan daftar pengguna yang terdaftar dalam grup "Users".

Meskipun proses *code*, *test*, *build*, dan *release* telah diotomatisasi, penelitian ini tidak menerapkan *continuous deployment*. Hal ini berarti bahwa *deployment* masih dilakukan secara manual, yang dapat memperlambat proses dan meningkatkan resiko kesalahan.

Selain itu, distribusi aplikasi menggunakan Firebase dirasa kurang optimal karena pengguna tidak dapat mengunduh aplikasi secara langsung. Pengguna harus menunggu pengembang mengirimkan link aplikasi terlebih dahulu. Hal ini dapat memperlambat proses pengunduhan dan pemasangan aplikasi.




**Gambar 13. Grup Users untuk Distribusi Aplikasi**

## 3.7. Operasi (*Operation*)

**Tabel 4. Hasil Operasi pada 3 Perangkat**

| Fitur | Perangkat | Status |
|---|---|---|
| Deteksi | Samsung Galaxy A50 | Berhasil |
| • Proses pemindaian berjalan dengan lancar dan memberikan hasil yang akurat | Xiaomi Redmi Note 9 | Berhasil |
| • Fitur deteksi melalui kamera berjalan dengan lancar. | Google Pixel 4A | Berhasil |
| • Fitur deteksi melalui galeri berjalan dengan lancar. | | |
| Tengkulak Beras | Samsung Galaxy A50 | Berhasil |
| • Informasi tengkulak beras tampil dengan benar. | Xiaomi Redmi Note 9 | Berhasil |
| • Fitur pencarian dapat melakukan *filter* berdasarkan masukan. | Google Pixel 4A | Berhasil |
| • Tombol "Hubungi Tengkulak" berfungsi dengan baik. | | |
| UI/UX | Samsung Galaxy A50 | Berhasil |
| • Antarmuka pengguna tidak ada kendala | Xiaomi Redmi Note 9 | Berhasil |
| • Tidak ada masalah visual atau navigasi yang terdeteksi. | Google Pixel 4A | Berhasil |
| Harga Beras | Samsung Galaxy A50 | Berhasil |
| • Informasi harga beras tampil dengan benar. | Xiaomi Redmi Note 9 | Berhasil |
| | Google Pixel 4A | Berhasil |

Pada tahap operasi, tim pengembang menggunakan aplikasi untuk mengevaluasi pengalaman pengguna dan memastikan kinerja aplikasi di berbagai perangkat. Praktik ini bertujuan untuk menguji keberlanjutan dan kinerja aplikasi dalam konteks penggunaan yang berbeda.

Hasil operasi aplikasi BERAS.AI pada beberapa fitur utama, diuji pada tiga perangkat yang berbeda ditunjukan pada Tabel 4.

Tabel 4 menunjukkan hasil yang positif pada berbagai perangkat, dapat disimpulkan bahwa aplikasi ini dapat memberikan pengalaman pengguna yang konsisten.

### 3.7. Pemantauan (*Monitoring*)

Tahap pemantauan dilakukan pada Firebase Perfomance untuk memperoleh informasi terkait peforma aplikasi dengan fokus pada 5 *metric* yang mencerminkan berbagai aspek peforma aplikasi [20].

1. *Metric App Start Time*

Metrik *App Start Time* mengukur waktu responsivitas aplikasi setelah dibuka [21], bervariasi antara 338ms hingga 1.61s. Waktu terlama mencapai 1.61s, mungkin dipengaruhi oleh kompleksitas aplikasi, ukuran *file*, dan koneksi jaringan yang kurang optimal.

2. *Metric Response Time*

Metrik ini mencatat waktu respons dari API setelah permintaan aplikasi [21]. Waktu respons tertinggi adalah 43.81s, terkait dengan fitur Harga dan Tengkulak yang melibatkan pengambilan data dari API. Penggunaan emulator menghasilkan waktu respons 43.81s, sedangkan perangkat fisik hanya memerlukan 3.97s, kemungkinan disebabkan oleh perbedaan performa.

3. *Metric Slow Rendering for MainActivity*

Metrik ini memuat persentase waktu saat aktivitas utama (MainActivity) mengalami penundaan dalam merender tampilan [21]. Dalam beberapa pengukuran terakhir, persentase slow rendering ini berkisar antara 0%, 50%, dan 100%.

4. *Metric Success Rate*

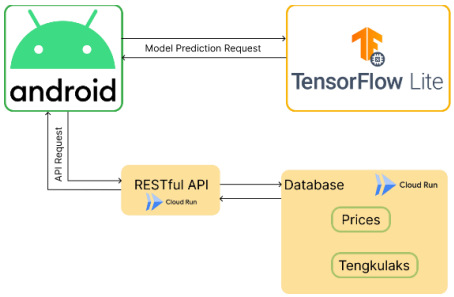
Metrik ini memuat persentase keberhasilan dari suatu tindakan atau proses tertentu [21]. Dalam proyek ini, persentase *success rate* yang konsisten pada 100% menunjukkan bahwa semua tindakan atau proses yang diukur berhasil tanpa adanya kegagalan atau kesalahan.

5. *Metric Response Payload Size*

Metrik ini mengukur ukuran data yang dikirimkan dari server ke aplikasi [21], dengan dua nilai payload size yang berbeda: 3.41KB dan 6.42KB. Payload size 3.41KB terkait dengan konten Harga, menunjukkan respons yang lebih kecil saat aplikasi meminta data dari server API untuk fitur tersebut. Sebaliknya, payload size 6.42KB terkait dengan konten fitur Tengkulak yang membutuhkan lebih banyak informasi dalam respons dari server.

### 3.8. Hasil Pengembangan Aplikasi Android

#### 1. Arsitektur Aplikasi




**Gambar 14. Diagram Arsitektur Aplikasi**

Aplikasi android ini terdiri dari arsitektur seperti pada Gambar 14. Aplikasi ini menggunakan TensorFlow Lite untuk memprediksi kualitas beras secara lokal pada perangkat. Gambar beras diambil melalui antarmuka pengguna aplikasi dan diproses oleh model TensorFlow Lite untuk menghasilkan prediksi kualitas beras. Selain itu, aplikasi juga memanfaatkan RESTful API yang di-*host* di *platform* Cloud Run untuk mengakses data eksternal seperti harga beras dan informasi tengkulak beras. Data ini disimpan dalam Cloud Firestore, sebuah database NoSQL. Aplikasi mengirim permintaan API ke server, yang kemudian mengambil data dari Firestore dan mengirimkannya kembali ke aplikasi untuk ditampilkan kepada pengguna.

#### 2. Pengujian Akurasi Deteksi

Hasil pengujian dengan gambar sample4.PNG menunjukkan model memberikan prediksi yang sangat percaya terhadap suatu kelas. Probabilitas prediksi untuk kelas "Tidak Layak" adalah 99%, menandakan keyakinan model bahwa gambar sample4.PNG termasuk dalam kelas "Tidak Layak". Waktu yang diperlukan untuk melakukan prediksi satu gambar adalah 44ms/step, menunjukkan kecepatan dan efisiensi model dalam pengujian. Waktu singkat ini menandakan bahwa model dapat bekerja dengan cepat dan dapat diimplementasikan secara *real-time* pada aplikasi seluler.

## 4. Kesimpulan

Analisis menunjukkan bahwa pengembangan aplikasi dengan pendekatan DevOps meningkatkan kualitas dan stabilitas aplikasi. Implementasi DevOps pada aplikasi pendeteksi kualitas beras memperbaiki proses pengembangan dengan otomatisasi pada tahap code, build, test, dan release. Sebanyak 60 pipeline telah dibuat sejak awal proyek, menunjukkan fleksibilitas pengembangan. Meskipun hasil aplikasi menunjukkan kualitas dan performa yang baik, disarankan untuk mengevaluasi teknologi terbaru dan mempertimbangkan *Continuous Deployment* dan mengevaluasi *platform* distribusi lain untuk memberikan gambaran yang lebih komprehensif tentang pilihan yang tersedia bagi pengembang.

## Referensi

[1]. T. S. Rathna Priya, A. R. L. Eliazer Nelson, K. Ravichandran, and U. Antony, "Nutritional and functional properties of coloured rice varieties of South India: a review," *J. Ethn. Food*, vol. 6, no. 1, p. 11, Oct. 2019, doi: 10.1186/s42779-019-0017-3.

[2]. W. M. Handani, N. Kusnadi, and D. Rachmina, "Prospek Swasembada Beras di Provinsi Kalimantan Timur," *J. Indones. Agribus*, vol. 9, no. 1, pp. 67–78, Jun. 2021, doi: 10.29244/jai.2021.9.1.67-78.

[3]. J. H. David and T. Kartinaty, "KARAKTERISTIK MUTU BERAS DI BERBAGAI PENGGILINGAN PADA SENTRA PADI DI KALIMANTAN BARAT," *JTAS*, vol. 3, no. 1, p. 276, Jun. 2019, doi: 10.35914/tabaro.v3i1.197.

[4]. Resa Setia Adiandri, S.TP, MP, Eka Rahayu, S.TP, M.Si, and Erwan Gustian Apriansyah, S.Sos, *Warta BSIP Pascapanen: Sertifikasi Mutu Beras Sebagai Pencegah Manipulasi Mutu*, Edisi Triwulan 2. 2023.

[5]. D. Lestari, N. Fadillah, and A. Ihsan, "Sistem Deteksi kualitas Beras Berdasarkan Warna menggunakan Fuzzy C-Means Clustering Guna Membantu Tingkat Pengetahuan Masyarakat," *InfoTekJar*, vol. 3, no. 2, pp. 32–38, Feb. 2019, doi: 10.30743/infotekjar.v3i2.920.

[6]. Intan Puspitasari, Harry J. Sumampouw, and Aneke Y. Punuindoong, "Pengaruh Kualitas Produk dan Kesesuaian Harga Terhadap Peningkatan Penjualan Beras Premium Pada Perum Bulog Divisi Regional Sulawesi Utara dan Gorontalo (Studi Kasus Pada Konsumen Wilayah Kota Manado)," vol. 6, no. 2, p. 60, 2018, doi: https://doi.org/10.35797/jab.v6.i002.%25p.

[7]. M. Mustofah and P. Utami, "Perangkat Penentu Kualitas Beras Ditinjau dari Kadar Air dan Berat Butir Menir Berbasis Arduino Uno," *ELINVO*, vol. 4, no. 1, pp. 39–48, Nov. 2019, doi: 10.21831/elinvo.v4i1.21516.

[8]. M. C. Custodio, R. P. Cuevas, J. Ynion, A. G. Laborte, M. L. Velasco, and M. Demont, "Rice quality: How is it defined by consumers, industry, food scientists, and geneticists?," *Trends in Food Science & Technology*, vol. 92, pp. 122–137, Oct. 2019, doi: 10.1016/j.tifs.2019.07.039.

[9]. Przemyslaw Gilski and Jacek Stefanski, "Android OS: A Review," *TEM Journal*, vol. 4, no. 1, pp. 116–120, 2015.

[10]. R. C. Dharmik, S. Chavhan, S. Gotarkar, and A. Pasoriya, "Rice quality analysis using image processing and machine learning," *3C TIC*, vol. 11, no. 2, pp. 158–164, Dec. 2022, doi: 10.17993/3ctic.2022.112.158-164.

[11]. M. S. Ardi, "Rancang Bangun Pendeteksi Kualitas Beras Menggunakan Metode K-Nearest Neighbor Berbasis Android," vol. 7, no. 2, 2021.

[12]. S. Alsaqqa, S. Sawalha, and H. Abdel-Nabi, "Agile Software Development: Methodologies and Trends," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 11, p. 246, Jul. 2020, doi: 10.3991/ijim.v14i11.13269.

[13]. A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Computer Science Review*, vol. 38, p. 100308, Nov. 2020, doi: 10.1016/j.cosrev.2020.100308.

[14]. R. Eramo, G. L. Scoccia, M. Nolletti, A. Celi, and M. Autili, "An Empirical Study on the Role of Devops in the Development of Mobile Applications," SSRN, preprint, 2024. doi: 10.2139/ssrn.4719199.

[15]. N. Azad and S. Hyrynsalmi, "DevOps critical success factors — A systematic literature review," *Information and Software Technology*, vol. 157, p. 107150, May 2023, doi: 10.1016/j.infsof.2023.107150.

[16]. M. O. Khan, "Fast Delivery, Continuously Build, Testing and Deployment with DevOps Pipeline Techniques on Cloud," *IJST*, vol. 13, no. 5, pp. 552–575, Feb. 2020, doi: 10.17485/ijst/2020/v13i05/148983.

[17]. Dr. Khurram Shahzad and Prof. David Williams, "CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY (CI/CD): AUTOMATING THE SOFTWARE DEVELOPMENT PIPELINE," *IJCST*, vol. 7, no. 4, Dec. 2023.

[18]. N. T. Phương Giang and T. T. Minh Khoa, "AUTOMATED CONTINUOUS INTEGRATION USING CIRCLECI AND FIREBASE FOR ANDROID APPLICATION DEVELOPMENT," *jst-iuh*, vol. 47, no. 05, Apr. 2021, doi: 10.46242/jst-iuh.v47i05.762.

[19]. S. Kaiser, Md. S. Haq, A. S. Tosun, and T. Korkmaz, "Container Technologies for ARM Architecture: A Comprehensive Survey of the State-of-the-Art," *IEEE Access*, vol. 10, pp. 84853–84881, 2022, doi: 10.1109/ACCESS.2022.3197151.

[20]. J. Harty, H. Zhang, L. Wei, L. Pascarella, M. Aniche, and W. Shang, "Logging Practices with Mobile Analytics: An Empirical Study on Firebase," in *2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft)*, Madrid, Spain: IEEE, May 2021, pp. 56–60. doi: 10.1109/MobileSoft52590.2021.00013.

[21]. Dario Piazza, "STARTUP PERFORMANCE ANALYSIS AND OPTIMIZATION OF AN ANDROID BANKING APPLICATION," Politecnico di Torino, Torino, North Italy, 2021.