Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 3, No. 4, April 2023, Hal. 151-164
DOI: https://doi.org/10.52436/1.jpti.285
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Rancang Bangun Aplikasi Quest Board Untuk Masyarakat Menggunakan Metode Devops Berbasis Android

Bima Adityo Kurniawan[*1], Acep Taryana[2], Yogi Ramadhani[3], Ari Fadli[4]

[1,2,3,4]Teknik Elektro, Universitas Jenderal Soedirman, Indonesia
Email: [1]badityo@gmail.com

Abstrak

Indonesia dikenal sebagai negara yang dikenal sangat menjunjung tinggi nilai gotong royong dan menolong satu sama lain. Dalam wilayah pedesaan masih sangat kental prinsip gotong royong ini. Namun dalam wilayah perkotaan, seringkali kita melihat masyarakat yang jangankan untuk dimintai pertolongan ataupun bantuan, sekadar berkenalan pun cukup sulit. Dikarenakan sifat individualistis yang lebih dominan pada penduduk kota dibandingkan penduduk desa, maka muncul sikap acuh tak acuh terhadap sesama. Berdasarkan masalah tersebut maka dikembangkan sebuah aplikasi bernama Kota Quest: Aplikasi Quest Board untuk masyarakat perkotaan. Pengembangan dan pembuatan aplikasi ini merupakan salah satu wujud penerapan kembali prinsip saling membantu satu sama lain bangsa Indonesia yang sudah mulai hilang khususnya pada daerah perkotaan. Aplikasi ini menggunakan metode *DevOps* dalam pengembangannya. *DevOps* merupakan metode baru dalam Software Development dan hadir sebagai solusi atas masalah metode modern lain dimana pengguna belum dapat merasakan produk yang sudah jadi selama tahap development. *DevOps* tidak terlalu mementingkan persyaratan sistem yang lengkap pada awal development. Sebagai gantinya, persyaratan dibuat secara langsung dalam bentuk model untuk kemudian diimplementasikan dan dikembangkan, sehingga pengguna dapat secara cepat mendapatkan gambaran umum dari produk yang akan dibuat. Implementasi pengembangan aplikasi ini menggunakan Kotlin sebagai bahasa pemrograman yang digunakan dan Firebase untuk back-end, serta beberapa tools lain yang mendukung pengembangan aplikasi. Hasil dari penelitian ini adalah berupa aplikasi mobile yang dapat dipasang pada perangkat Android dan terhubung dengan Firebase datastore. Aplikasi ini diharapkan dapat membantu melestarikan budaya gotong royong yang sudah sulit ditemukan pada masyarakat kota.

Kata kunci: *Aplikasi Android, DevOps, Quest Board*

Design and Build a Quest Board Application for the Community using the Android-Based Devops Method

Abstract

Indonesia is known as a nation that upholds the value of collaborative work and helping each other. Especially in rural areas, these principles are still held tightly. However, in urban areas, we often see how hard it is to ask for help from big city citizens, let alone befriend them. Because of this dominant individualistic behavior from city dwellers as opposed to rural villagers, an ignorant mindset is formed within most people there. Based on this problem, Kota Quest: Quest Board Application for City Folks is created. The development of this Application is an effort to re-apply the Indonesian principles to help each other, especially in urban areas. The Application utilizes the DevOps method in its development cycle. DevOps is a new method in Software Development that presents itself as a solution to other modern techniques in which users cannot feel the finished product during development cycles. DevOps does not emphasize detailed system requirements in the early development steps. Instead, the conditions are made directly using models to be implemented and developed, after which users can grasp the big picture of the finished product. The Application implements Kotlin as the programming language, Firebase as the back end, and other tools to develop the Application. The result is a mobile Android application that connects directly to Firebase datastore in real time. Hopefully, this Application will preserve the communal work culture, which has been hard to find among city dwellers.

Keywords: *Android Application, DevOps, Quest Board*

1. PENDAHULUAN

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Indonesia dikenal sebagai negara bermasyarakat Indonesia dikenal sebagai negara bermasyarakat majemuk yang terdiri dari berbagai suku, adat istiadat, agama dan kepercayaan yang berbeda- beda. Keberagaman ini kemudian menciptakan sebuah tradisi masyarakat yang lekat dalam berbagai aspek kehidupan sehari-hari. Salah satunya adalah tradisi gotong royong. Gotong royong sendiri merupakan bentuk kerja sama kelompok masyarakat untuk mencapai hasil positif tanpa memikirkan dan mengutamakan keuntungan bagi salah satu individu atau kelompok saja, melainkan untuk kebahagian bersama. Budaya ini memiliki nilai moral yang baik dalam kehidupan masyarakat [1]. Indonesia juga dikenal sebagai negara yang dikenal sangat menjunjung tinggi nilai gotong royong dan menolong satu sama lain. Dalam wilayah pedesaan masih sangat kental prinsip gotong royong ini. Namun dalam wilayah perkotaan, seringkali kita melihat masyarakat yang jangankan untuk dimintai pertolongan ataupun bantuan, sekadar berkenalan pun cukup sulit. Dikarenakan sifat individualistis yang lebih dominan pada penduduk kota dibandingkan penduduk desa, maka muncul sikap acuh tak acuh terhadap sesama. Orang Indonesia dikenal suka tolong menolong, namun lebih terlihat pada orang desa dibanding orang kota. Dikarenakan sifat individualistis yang lebih dominan pada penduduk kota dibandingkan penduduk desa, maka muncul sikap acuh tak acuh terhadap sesama. Studi yang dilakukan Nottingham Trent University menyatakan setiap orang rata rata memeriksa ponsel mereka 85 kali dalam sehari [2]. Perhatian masyarakat lebih tertuju pada ponsel dibanding lingkungan sekitarnya.

Dalam pengembangan suatu perangkat lunak khususnya aplikasi terdapat beberapa hal yang harus diperhatikan yaitu, sumber daya manusia yang terkait dalam pengembangan perangkat lunak, estimasi biaya dan metode yang diterapkan dalam proses pengembangan aplikasi tersebut. Terdapat banyak metode dalam proses pengembangan perangkat lunak salah satunya yaitu DevOps. Aplikasi ini menerapkan metode pendekatan DevOps. Pendekatan DevOps kurang lebih mirip dengan pendekatan Agile. Bahkan DevOps bisa dikategorikan sebagai pendekatan implementasi dari metodologi Agile. DevOps merupakan ekstensi dari Agile dengan tambahan otomasi produk untuk meningkatkan kolaborasi antara tim development dengan operations[3]. Aplikasi Quest Board sendiri mendasari prinsip kerjanya sebagai papan berisi quest yang dapat digunakan oleh masyarakat sebagai sarana memberitahu masalah mereka kemudian mencari siapa yang dapat membantu menyelesaikan permintaan atau masalah yang mereka publikasikan.

Beberapa penelitian terkait quest board ini diantaranya yaitu Penelitian untuk memetakan jalur pendakian yang ada Kawasan Hutan Lindung Bukit Cemara Geseng Via Desa Silangjana menggunakan aplikasi Gps *Alphine Quest* Dan *Google Earth Pro* dengan cara survei lapangan dan wawancara terkait dengan kearifan lokal masyarakat. hasil penelitian ini memberikan gambaran tentang (1) Akses menuju titik start pendakian,(2) Tata cara perijinan pendakian. (3) Lokasi objek suci dan Diskripsi tentang jalur pendakian Objek suci yang berupa *Sanggah*, *Pelinggih* Pohon besar, dan *Pura* di Puncak.(4) Kepercayaan Lokal, berbentuk *awig-awig* yaitu "*Sampunang Ngomong Capek*", karena mengucapkan kata tersebut dapat memberikan energi negatif [4]. Selanjutnya, penelitian yang dilakukan yaitu Pengembangan Aplikasi Daily Quest: Aplikasi Untuk Menangani Kemalasan Pada Anak Menggunakan Platform Android, hasil pengujian validasi, usabilitas, dan kompatibilitas menunjukan aplikasi Daily Quest. Pada pengujian validasi menghasilkan tingkat keberhasilan 100%. Pengujian usabilitas menghasilkan *success rate* sebesar 82,8% pada sistem orang tua dan 91,6% pada sistem anak. Pengujian dengan metode *system usability scale* (SUS) menghasilkan nilai rata-rata sebesar 66 pada sistem orang tua, golongkan ke dalam kategori *grade* C yaitu *marginal* yakni berada diantara dapat diterima dan tidak. Sedangkan pada sistem anak menghasilkan nilai rata-rata sebesar 73, digolongkan ke dalam kategori *grade* B-yaitu *acceptable* yakni dapat diterima. Pengujian kompatibilitas sistem, aplikasi Daily Quest hanya dapat berjalan dengan baik pada versi android lollipop 5.0 dengan tingkatan API minimal 21 [5].

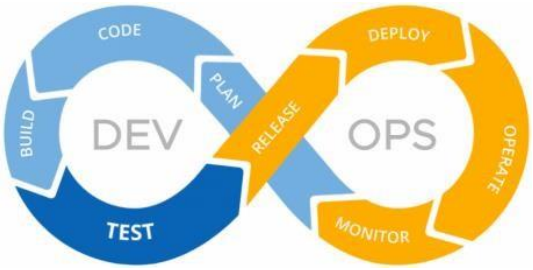
Penelitian lain yaitu bertujuan menemukan karakteristik item evaluasi pada mata pelajaran Pendidikan Agama Islam. Data yang digunakan untuk analisa seluruh peserta didik kelas VI dengan jumlah 56 peserta didik dan jumlah soal 40 item. Analisa data dengan menggunakan Program QUEST untuk mendapatkan karakteristik dari item soal evaluasi. Hasil penelitian menunjukkan bahwa sebanyak 5 (lima) item terjadi item has perfect score, sebagian besar item memiliki tingkat kesukaran "Sedang". Sebanyak 48,5% item memiliki daya beda dengan kategori "sangat Baik". Sebanyak 88,5% item merupakan item yang sesuai untuk mengevaluasi kemampuan peserta didik pada mata pelajaran Pendidikan Agama Islam. Estimasi reliabilitas sebesar 0,75, dan jumlah distractor yang efektif sebanyak 50% [6]. Berdasarkan latar belakang tersebut, maka pada tulisan ini membahas tentang Rancang Bangun Aplikasi Quest Board Untuk Masyarakat Menggunakan Metode DevOps Berbasis Android.

2. METODE PENELITIAN

Metode Metode yang digunakan dalam penelitian ini adalah metode DevOps. DevOps terdiri dari Perancangan (Development) dan Pengoperasian (Operations). Tim developer memiliki tanggung jawab untuk melaksanakan perencanaan (plan), membuat program aplikasi (code, build, test), merilis program aplikasi

p-ISSN: 2775-4227
e-ISSN: 2775-4219

(release), dan menyebarkan aplikasi (deploy). Sedangkan tim pengoperasi memiliki tanggung jawab untuk mengoperasikan (operate), dan melaksanakan monitoring (monitor). Setiap tanggung jawab perkerjaan tersebut dikerjakan secara berurutan, mulai dari perencanaan sampai dengan monitoring untuk setiap siklusnya. DevOps yang terotomatisasi mengeksekusi siklus dengan cepat dan sering untuk mengantisipasi perubahan kebutuhan baru agar sesuai dengan kebutuhan customer [7].



Gambar 1. Siklus DevOps.

2.1. Perencanaan (Plan)

Mempersiapkan langkah-langkah awal yang dibutuhkan dalam melakukan penelitian, mulai dari identifikasi masalah sampai pembuatan aplikasi. Perancanaan sistem menggunakan UML (Unified Modelling Language). Adapun UML yang digunakan adalah Use Case Diagram untuk menjelaskan hubungan pengguna dengan sistem pada aplikasi, activity diagram yang menjelaskan alur proses pada suatu use case aplikasi, sequence diagram yang menjelaskan proses aplikasi secara sekuensial dan bertahap, ada pula class diagram yang akan menggambarkan berbagai class pada aplikasi serta hubungannya dengan satu sama lain.

2.2. Pemrograman (Code)

Untuk merealisasikan tahap coding ini dilakukan pengembangan aplikasi menggunakan Android Studio sebagai IDE untuk menulis source code pada program yang akan digunakan.

2.3. Pembangunan (Build)

Tahap ini menggunakan Gradle sebagai alat untuk melakukan build secara otomatis. Build di sini termasuk ke dalamnya sebagai aktivitas melakukan compile dan packaging. Sehingga tidak perlu lagi mengkompilasikan source code Kotlin menggunakan command line.

2.4. Pengujian (Testing)

Pada tahap testing, uji coba dilakukan pada aplikasi dengan menjalankannya di Android Virtual Device Manager pada Android Studio menggunakan tipe smartphone dengan sistem operasi Android dan ukuran resolusi layar yang berbeda-beda.

2.5. Perilisan (Release)

Setelah pengujian berhasil, aplikasi akan dirilis dalam bentuk .apk dan dapat dipasang pada smartphone sungguhan (bukan berbentuk emulator pada komputer). Untuk release disini berupa menyimpan folder source code aplikasi ke sebuah repository yang nantinya akan dirilis menggunakan Github.

2.6. Peluncuran (Deploy)

Firebase digunakan untuk melakukan deployment terhadap aplikasi yang telah dirilis. Firebase sudah dapat diandalkan sebagai bagian back-end dari aplikasi ini.

2.7. Pengoperasian (Operate)

Fase inti Operations dari DevOps adalah memastikan pengoperasian aplikasi berjalan dengan lancar. Aplikasi harus dapat menyesuaikan dengan lonjakan pengguna yang menggunakannya. Data yang masuk dan keluar dari Firestore harus dipastikan sesuai dengan yang ada pada tampilan aplikasi.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

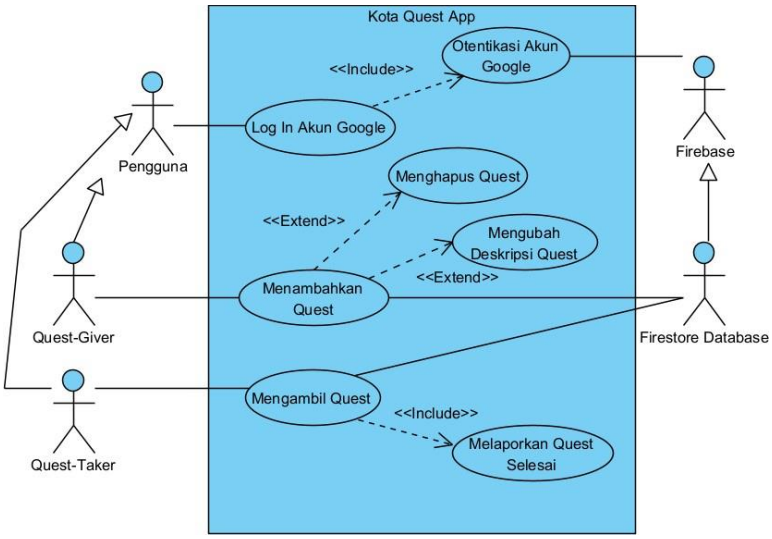
2.8. Pemantauan (Monitor)

Monitoring yang dilakukan pada aplikasi Quest Board untuk masyarakat ini menggunakan Analytics dari Firebase pada Cloud Firestore Database untuk mengukur masuk keluarnya data melalui aplikasi.

3. HASIL DAN PEMBAHASAN

3.1. Perencanaan (Plan)

Tahap perencanaan menggunakan berbagai macam diagram untuk menggambarkan aplikasi yang akan dibuat. Use Case Diagram penelitian ini melibatkan sistem dengan pengguna sebagai penerima dan penyedia informasi. Activity Diagram untuk menjelaskan urutan aktivitas dalam suatu proses. Sequence Diagram menampilkan interaksi antar objek dalam dua dimensi. Sedangkan Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas



Gambar 2. Use Case Diagram Kota Quest

Dari Use Case Diagram di atas dapat dilihat terdapat Terdapat dua aktor penting yang berperan sebagai inti dari berjalannya sistem dalam aplikasi Quest Board atau Kota Quest, yaitu Quest Giver dan Quest Taker. Keduanya merupakan generaliasi dari aktor pengguna aplikasi. Sebelum dapat menggunakan aplikasi Kota Quest, pengguna harus log in akun Google terlebih dahulu dimana use case tersebut sudah termasuk di dalamnya otentikasi akun Google yang dilakukan oleh Firebase. Quest Giver disini merupakan orang yang memiliki permasalahan yang ingin dipublikasikan dan meminta tolong untuk dibantu oleh orang lain dalam menyelesaikannya. Sehingga Quest Giver lah yang berperan dalam menambahkan Quest dalam aplikasi ini. Selain memberikan Quest, Quest-Giver juga dapat menghapus dan mengubah deskripsi dalam Quest tersebut. Ketika quest sudah dimasukkan ke dalam aplikasi, maka tinggal menunggu pengguna lain, dalam hal ini Quest Taker untuk mengambilnya. Setelah quest diambil, maka kedua aktor dapat melihat status dari quest tersebut apakah sudah diselesaikan atau belum. Quest Taker kemudian dapat melaporkan selesai atau tidaknya quest yang telah diambil.

Untuk Use Case yang akan dibahas lebih lanjut pada Activity Diagram dan Sequence Diagram hanya mengambil 3 Use Case saja yaitu:

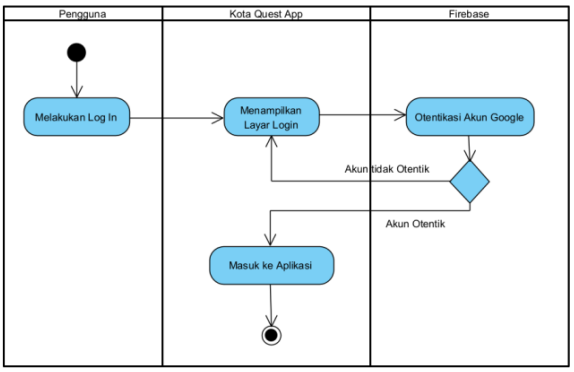
1. Log In Akun Google
2. Menambahkan Quest
3. Mengambil Quest.

3.1.1. Activity Diagram

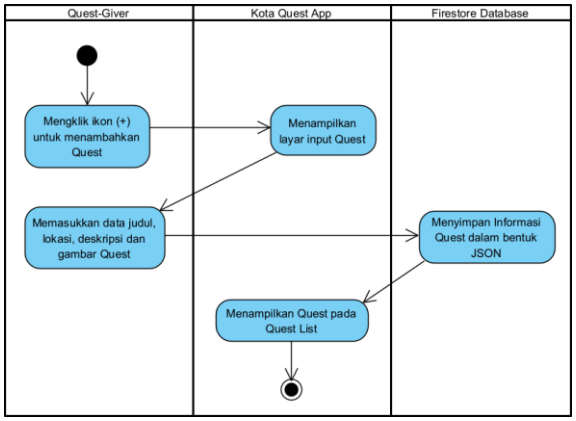
Activity Diagram Aplikasi pada gambar 3 memodelkan bagaimana proses yang terjadi ketika pengguna hendak masuk ke aplikasi menggunakan akun *Google*. Aktivitas yang dilakukan oleh pengguna yang pertama yaitu melakukan *Log In*, kemudian aplikasi akan menampilkan layar *login* untuk pengguna memilih akun *Google* yang bisa dipakai untuk masuk. Setelah memilih akun tersebut, *Firebase* akan mengotentikasi apakah akun

p-ISSN: 2775-4227
e-ISSN: 2775-4219

memang otentik atau tidak. Jika tidak maka pengguna akan dikembalikan ke layar *log in*. Jika akun benar maka pengguna akan dibawa masuk ke aplikasi.

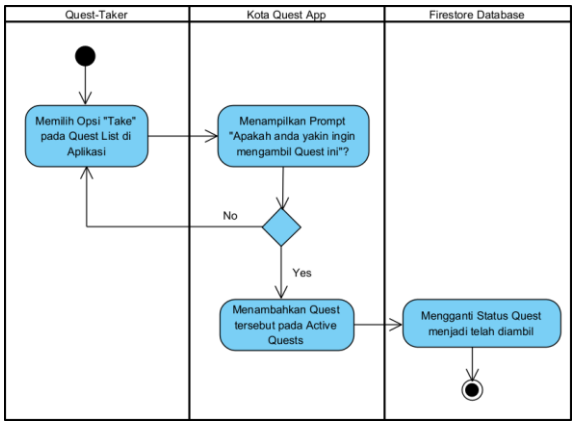


Gambar 3. Activity Diagram Login



Gambar 4. Activity Diagram Menambahkan Quest

Activity Diagram pada gambar 4 memodelkan bagaimana seorang *Quest-Giver* menambahkan *quest* pada Kota Quest. Pertama yang harus dilakukan adalah mengklik ikon (+) pada aplikasi, kemudian akan ditampilkan layar input *quest* dari aplikasi. *Quest Giver* memasukkan data judul, lokasi, deskripsi dan gambar *quest* untuk kemudian informasi tersebut dikirim dan disimpan dalam bentuk *file* JSON pada *Firestore database*. Terakhir aplikasi akan menampilkan *Quest* tersebut pada *Quest list*



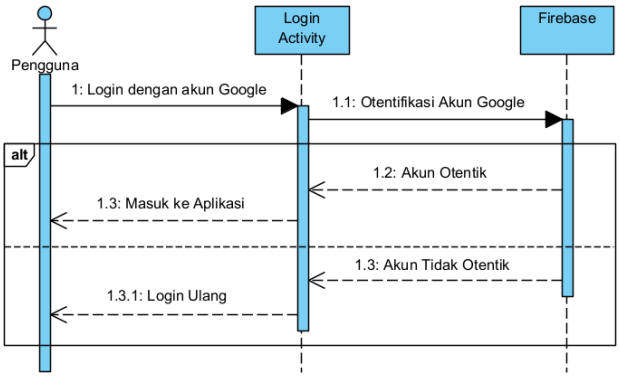
Gambar 5. Activity Diagram Mengambil Quest

Activity Diagram pada gambar 5 memodelkan bagaimana seorang *Quest-Taker* mengambil *quest* pada Kota Quest. Pertama yang harus dilakukan adalah memilih opsi *Take* pada *Quest List* di aplikasi, kemudian akan ditampilkan sebuah *prompt* "Apakah anda yakin ingin mengambil *quest* ini?". Jika dibalas tidak, maka akan kembali ke awal, jika dibalas ya, aplikasi akan menambahkan *Quest* tersebut ke *Active Quests* pengguna

p-ISSN: 2775-4227
e-ISSN: 2775-4219

tersebut. Terakhir *Firestore Database* mengubah status *quest* tersebut menjadi sudah diambil sehingga disembunyikan dari *Quest List* aplikasi.

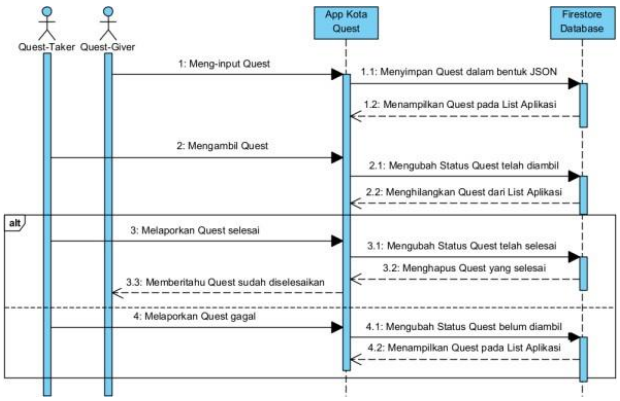
3.1.2. Sequence Diagram



Gambar 6. Sequence Diagram Login

Sequence Diagram pada gambar 6 memodelkan bagaimana proses yang terjadi selama *log in*. Aktor pengguna akan mengirimkan pesan untuk *login* dengan akun *Google* yang akan dikirimkan kepada *Login Activity*, kemudian dikirimkan lagi sebagai otentikasi akun *Google* ke *Firebase*. Ada dua skenario yang akan terjadi. Pertama jika akun benar otentik maka *Firebase* akan mengirimkan balasan pesan bahwa akun otentik dan aplikasi akan memberi balasan pesan ke pengguna untuk masuk ke aplikasi, jika tidak maka aplikasi akan memberi balasan pesan untuk melakukan *login* ulang.

Sequence Diagram pada gambar 7 memodelkan bagaimana proses yang terjadi selama pengguna (dalam hal ini *Quest-Giver* dan *Quest-Taker*) menambah dan mengambil *Quest*. *Sequence Diagram* ini bisa disebut sebagai prinsip kerja dari aplikasi Kota Quest. Pertama aktor *Ques- Giver* akan mengirim pesan untuk meng-input *Quest* ke aplikasi. Aplikasi Kota Quest kemudian akan mengirim pesan kepada *Firestore Database* untuk menyimpan informasi q*uest* yang telah dimasukkan. *Firestore Database* akan memberikan balasan pesan untuk menampilkan *quest* pada list aplikasi dengan mengambil data informasi *quest* yang sudah disimpan.



Gambar 7. Sequence Diagram Menambah dan Mengambil Quest.

Selanjutnya, aktor kedua yaitu *Quest-Taker* akan mengirim pesan untuk mengambil *quest* tersebut kepada aplikasi. Lalu diberikan pesan lanjutan kepada *firestore database* untuk mengubah status *quest* menjadi sedang diambil. *Firestore* lalu akan memberi pesan balasan untuk menyembunyikan *quest* dari aplikasi karena sudah *in progress*. Setelah itu ada 2 skenario alternatif yang dapat terjadi. Yang pertama jika *Quest-Taker* memberi pesan bahwa *quest* berhasil diselesaikan, Aplikasi akan mengirimkan pesan kepada *firestore* untuk mengubah status *quest* menjadi sudah selesai, kemudian *quest* tersebut akan dihapus dari *firestore database* dan aplikasi. App Kota Quest akan memberi tahu kepada *Quest-Giver* bahwa *quest* sudah diselesaikan. Yang kedua jika *Quest-Taker* memberi pesan bahwa *quest* gagal diselesaikan (*drop*), Aplikasi akan mengirimkan pesan kepada *firestore* untuk mengubah status *quest* menjadi belum selesai, kemudian *quest* tersebut akan ditampilkan kembali pada aplikasi untuk diambil oleh pengguna lain

p-ISSN: 2775-4227
e-ISSN: 2775-4219

3.1.3. Class Diagram

Ada 8 *Class* dalam aplikasi ini, yaitu:

1.) *ActivitySplash*
Class ini merupakan *loading screen* awal ketika aplikasi dibuka.

2.) *ActivityLogin*
Class ini adalah *login page* sebelum pengguna dapat menggunakan aplikasi. *Class* ini memiliki asosiasi dengan *ActivitySplash*. *Class* ini memiliki hubungan agregasi dengan *MainActivity* dimana walaupun *ActivityLogin* dihilangkan, *MainActivity* masih dapat dijalankan tanpa masalah.

3.) *MainActivity*
Sesuai dengan namanya, *MainActivity* bisa dibilang sebagai *class* induk aplikasi yang akan menampung fungsi penting terutama untuk hubungan input data ke *Firebase*. Merupakan generalisasi dari *MainActivity*.

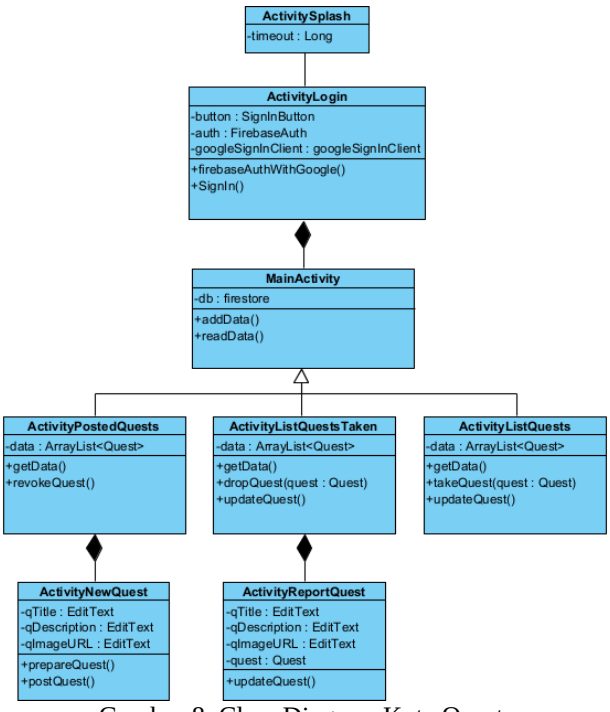
4.) *ActivityListQuests*
Class ini menampilkan *Quests* yang ada dengan tipe data *array* yang akan menampung data elemen *quest* tersebut. Terdapat 2 fungsi penting disini yaitu *takeQuest(quest: Quest*) untuk mengambilnya dan *updateQuest*() yang akan memperbaharui dan melakukan sinkronisasi data dengan *Database Firestore* yang ada pada *Firebase*. Merupakan generalisasi dari *MainActivity*.

5.) *ActivityListQuestsTaken*
Kurang lebih sama seperti *ActivityListQuest* hanya saja *class* ini memiliki fungsi *dropQuest(quest: Quest)* sebagai pilihan untuk pengguna apabila tidak sanggup menyelesaikan *Quest* yang sudah diambil untuk kemudian akan diperbaharui statusnya di *database Firestore* menjadi "belum diambil" dan akan dimunculkan kembali pada *ActivityListQuests*. Merupakan generalisasi dari *MainActivity*.

6.) *ActivityPostedQuests*
Merupakan *class* untuk pengguna yang memberikan *quest*. Dari fungsi *revokeQuest (quest: Quest)* maka pengguna *Quest Giver* dapat menarik kembali *quest* yang belum diambil.

7.) *ActivityNewQuest*
Memungkinkan pengguna untuk membuat *quest* baru yang akan ditampilkan pada *ActivityQuestList*. Judul, deskripsi dan gambar *quest* dapat dibuat dengan *attribute* yang sesuai, dalam hal ini *qTitle*, qDescription, keduanya dengan tipe data *EditText* dan *qImageURL* dengan tipe data *String*. Berketergantungan (*dependent*) terhadap *ActivityPostedQuests*.

8.) *ActivityReportQuest*
Memungkinkan pengguna yang mengambil *quest* (*Quest Taker*) untuk melaporkan *quest* yang sudah diselesaikan beserta dengan bukti deskripsi dan gambar. Berketergantungan (*dependent*) terhadap *ActivityListQuestsTaken*

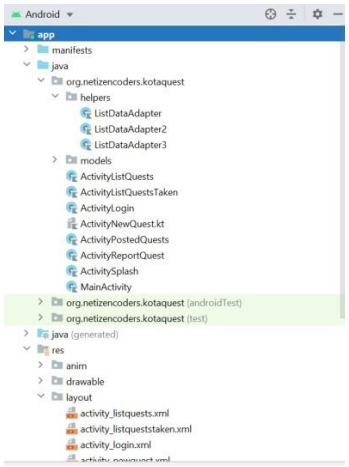


Gambar 8. Class Diagram Kota Quest.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

3.2. Pemrograman (Code)

Pembuatan aplikasi untuk perangkat Android dilakukan setelah perencanaan sudah dibuat. Aplikasi ditulis dalam Bahasa Kotlin yang menggunakan gradle sebagai komponen utama compiler & builder dalam aplikasi ini. Penulisan kode program (coding) disini adalah bagaimana cara mengembangkan hasil analisa dan perancangan yang telah dilakukan menjadi suatu sistem yang utuh. Ada 3 folder utama yang akan dibahas dalam struktur proyek aplikasi Kota Quest. Yaitu folder models, helpers dan layout.



Gambar 9. Struktur Project Kota Quest

Inti dari aplikasi ini ada di dalam *folder models* yang memuat seluruh *Activities* aplikasi yang di dalamnya berupa variabel, fungsi dan *library* yang akan digunakan untuk dapat menjalankan aplikasi Kota Quest. *Activities* pada aplikasi *Android* disini merujuk pada *state* ketika suatu kondisi terpenuhi untuk menampilkan antarmuka tersebut. Misalnya *MainActivity.kt* bisa disebut sebagai fungsi utama dari aplikasi yang dibuat. Yang pertama adalah untuk mengimpor *library* utama yang akan digunakan. Pada awal *Source Code MainActivity.kt*:

```
import android.content.Intent //launchactivity
import android.os.Bundle //used to passdata between activities
import android.util.Log //API for sendinglog output
import android.widget.TextView //A user interface element that displays text to
theuser
import androidx.appcompat.app.AppCompatActivity
//Base class for activities that wish to use some of the newer platform features
onolder Android devices
import com.google.android.material.bottomnavigation.BottomNavigationView
//Represents a standard bottom navigation bar for application
import com.google.firebase.firestore.ktx.firestore
//connect to Firestore
import com.google.firebase.ktx.Firebase
//connect to Firebase
```

MainActivity.kt merupakan *Source Code* pertama yang akan pertama dieksekusi oleh aplikasi saat pengguna pertama membukanya. Semua aplikasi membutuhkan *android.content.Intent* dibuat untuk bisa meluncurkan aplikasi, *android.os.Bundle* digunakan untuk menyalurkan data antar *Activity*. *Android.util.Log* merupakan API (*Application Programming Interface*) yang berisi definisi protokol untuk mengirimkan output log. *Android.widget.TextView* merupakan elemen antarmuka yang menampilkan teks kepada pengguna. *Android.x.app.compat.app.AppCompatActivity* berfungsi sebagai *class* dasar untuk *activity* agar bisa menggunakan fitur platform terbaru pada perangkat *Android* versi lama. *BottomNavigation* sesuai namanya adalah untuk navigasi aplikasi di posisi bawah layar. Library *ktx.firestore* untuk dapat mengakses *database Firestore*, sedangkan *ktx.firebase* untuk dapat mengakses fitur-fitur dari *Firebase*. Tetapi selain butuh *library* tersebut juga perlu dituliskan perintah berupa: private val db. = Firebase.*firestore* untuk dapat mengakses *firestore database*. Dalam pengembangan *Android*, setiap ingin menampilkan *list* dari kumpulan *item* secara vertikal, maka digunakan ListViewHolder yang sudah memiliki data terpopulasi menggunakan *Adapter*. *Adapter* disini berfungsi untuk mengkonversi data dari elemen *ArrayList<Quest>* yang berisi *Quest* menjadi elemen yang dapat ditampilkan pada layar antarmuka sesuai format yang digunakan.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Gambar 10. Hubungan Adapter-helper pada Kota Quest

Aplikasi *Quest Board* menggunakan *Adapter View* dengan tipe *RecyclerView*. *RecyclerView* disini merupakan perbaikan dari *ListView* dimana *list* yang dibuat akan menggunakan kembali *cells* ketika pengguna melakukan *scroll up/down* aplikasi. Maka karena ini akan lebih memungkinkan untuk mengimplementasikan *View Holder* pada *adapter*, yangmana sudah menjadi *default* pada *RecyclerView*. Hal ini juga berarti penggunaan *RecyclerView* dapat melepaskan *list item* yang akan ditampilkan dari *container*-nya sehingga penempatan *item* akan lebih mudah saat *runtime* dengan *layout* yang berbeda- beda. Kemudian dengan *RecyclerView*, animasi yang digunakan untuk *list* akan terpisah dan dimasukkan ke dalam *ItemAnimator* jika ada. *RecyclerView* memberikan kontrol yang fleksibel dalam mengurus data berupa *list* yang mengikuti prinsip kerja seperti namanya, yaitu penggunaan ulang. *Source Code* aplikasi ini dapat dilihat pada lampiran serta tersedia pada: https://github.com/ditocpp/Kota-Quest. *Layout source code* ditulis dalam bahasa XML yang akan menjadi tampilan yang bisa dilihat oleh pengguna saat aplikasi dijalankan. Dengan kata lain, antarmuka yang akan ditampilkan Kota Quest berada pada *file-file* di *folder* ini dan berhubungan erat dengan *activity* pada *folder models*.

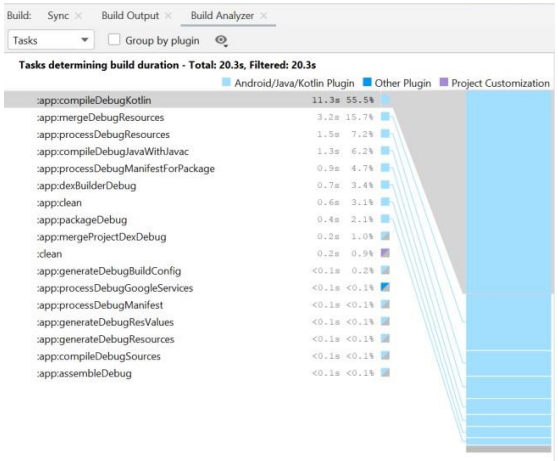
3.3. Pembangunan (Build)

Peluncuran aplikasi Kota Quest berbeda dengan perilisan aplikasi pada tahap kelima dari *DevOps* dimana pada perilisan bersifat mempublikasikan *build* aplikasi yang sudah dites dapat berjalan namun sebenarnya belum siap masuk dalam produksi dan penyebaran massal kepada pengguna. Lebih tepatnya *release* menandakan pembaharuan versi dari aplikasi. yang belum disebarluaskan melalui tahap *deployment* ini.

Setelah kode sumber dibuat pada IDE *Android Studio*, maka dapat dilakukan *build* aplikasi dengan menggunakan *gradle*. Proses *build* ini melibatkan banyak alat dan proses yang akan mengubah proyek aplikasi yang sudah dibuat ke dalam bentuk *Android Application Package* (.apk).

Proses *build* pada aplikasi ini sesuai pada tahap- tahap di bawah ini:

1. *Compiler* mengkonversikan *Source Code* yang telah dibuat ke dalam format *file* DEX (*Dalvik Executable*), di dalamnya terdapat *code* yang dapat berjalan pada perangkat *Android*.
2. *Packager* mengkombinasikan *file* DEX tersebut kemudian mengkompilasikannya sebagai *resource* terkompilasi dan menjadi APK. Sebelum aplikasi ini dapat diinstall pada perangkat *Android*, APK harus mendapatkan *sign* untuk *keystore* agar dapat dirilis.
3. *Packager* kemudian akan melakukan *signing* pada APK yang telah dibuat menggunakan *keystore debug/release*.



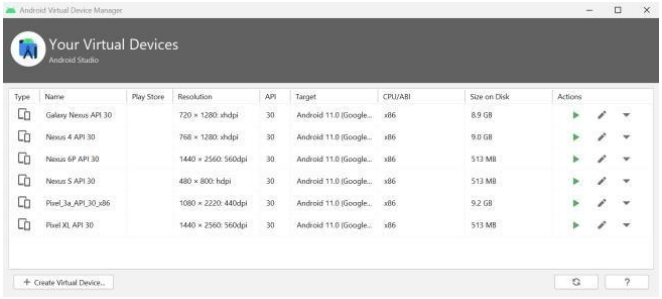
Gambar 11. Tampilan Build Analyzer

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Dengan menggunakan *Build* Analyzer dari *Android* Studio, didapatkan persentase waktu yang dihabiskan selama proses *build.* Total durasi 22 detik, dengan 11.3 detik (55%) waktu *build* tersebut dihabiskan untuk *app:compileDebugKotlin* yang merupakan proses input-output pada aplikasi. Setelah dilakukan build tersebut maka aplikasi sudah memilik output file APK yang bisa digunakan untuk uji coba menjalankannya pada *smartphone* untuk diuji coba.

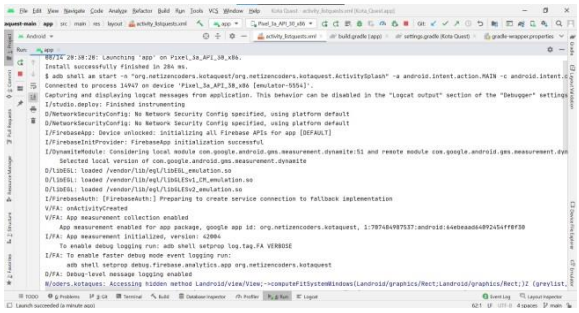
3.4. Pengujian (Test)

Pengujian pada aplikasi Kota Quest dilakukan secara internal pada *emulator* yang sudah terintegrasi dengan *Android Studio.* Untuk uji coba aplikasi akan dilakukan pada 6 buah perangkat *smartphone* menggunakan *Android Virtual Device Manager* sebagai pengujian terinstrumentasi terhadap aplikasi Kota Quest. Sistem operasi *Android* yang digunakan pada setiap perangkat ini bervarian dari *Lollipop, Nougat, Oreo, Pie, Q dan R*.

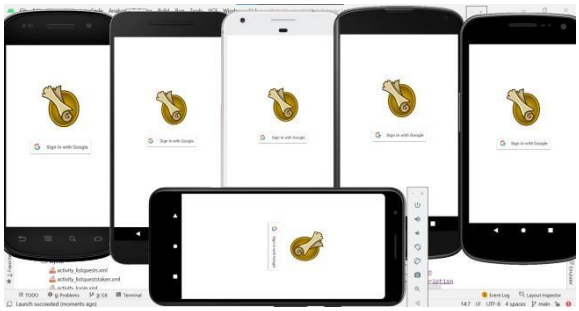


Gambar 12 Tampilan 6 Test Smartphone Android Virtual Device Manager

Sesuai dengan hasil *console* pada gambar, aplikasi berjalan dengan lancar tanpa ditemukan masalah yang signifikan. *Login page* bisa ditampilkan pada *emulator Android Studio* namun secara fungsionalitas masih belum bisa melanjutkan karena aplikasi ini belum dirilis dan belum dihubungkan dengan *Firebase*, karena pada dasarnya semua prinsip kerja aplikasi ini membutuhkan fitur- fitur *back-end* yang disediakan pada *Firebase*. Hasil *testing* berjalan dengan lancar dan membuktikan bahwa aplikasi dapat berjalan pada keenam perangkat sampel.


Gambar 13 console emulation run Kota Quest pada Android Studio

Dari gambar 4.16 dapat dilihat bahwa aplikasi Kota Quest telah lolos pengujian untuk dapat setidaknya menginisialisasikan aplikasinya pada 6 jenis *smartphone* dengan resolusi dan versi *Android* yang berbeda-beda. Pada tabel 4.4 diperlihatkan hasil dari ujicoba tersebut dengan urutan pada gambar yaitu dari kiri ke kanan


Gambar 14. Tampilan Login Page pada Android Virtual Device

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Kemudian yang perlu diperhatikan disini adalah karena aplikasi ini memang tidak memiliki desain tampilan untuk ketika *smartphone* dalam keadaan tidur (*landscape*), maka saat ujicoba di Galaxy Nexus tetap menampilkan aplikasi dengan tampilan berdiri (*potrait*). Dari tahap ujicoba ini didapatkan bahwa ternyata dalam pelaksanaan pengembangan aplikasi *Android* yang menggunakan *Firebase* dengan metode *DevOps* akan mengalami permasalahan, yaitu aplikasi yang menggunakan *Firebase* belum begitu pantas untuk diuji kelayakannya karena agar dapat berfungsi membutuhkan fitur teknologi pada *Firebase.* Hal ini terjadi dikarenakan *Firebase* sendiri termasuk ke dalam tahap keenam dalam *DevOps* yakni *Deploy.* Sehingga sejatinya aplikasi ini baru dapat diuji secara maksimal pada tahap *Operate*. Untuk menanggulanginya maka pada penelitian tahap *test* hanya dilakukan uji kelancaran aplikasi pada 6 *smartphone* dengan versi *Android* OS yang berbeda-beda.

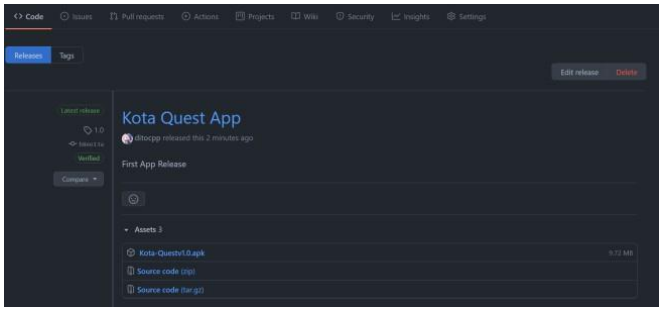
Tabel 1. Hasil Uji Coba Aplikasi pada *Android Virtual Device*

No	Nama *Smartphone*	Versi OS	Hasil
1.	Galaxy Nexus	*R*	Lancar
2.	Nexus 4	*Q*	Lancar
3.	Nexus 6P	*Pie*	Lancar
4.	Nexus S	*Oreo*	Lancar
5.	Pixel 3A	*Nougat*	Lancar
6.	Pixel XL (*Landscape*)	*Marshmellow*	Lancar

.

3.5. Perilisan (Release)

Fase perilisan aplikasi disini merupakan akhir dari tahap *development* pada *DevOps*, dimana aplikasi sudah siap untuk memasuki tahap *operations* yang akan menyiapkan aplikasi untuk siap terintegrasi dalam lingkungan produksi. Saat sudah memasuki tahap ini, aplikasi sudah menjalani ujicoba pada tahap sebelumnya sehingga sudah dapat melanjutkan ke tahap operasional. Walau pada realisasinya tahap *release* bukan berarti aplikasi sudah dapat dipublikasikan kepada pengguna, maka versi rilis aplikasi pada tahap ini lebih dikhususkan sebagai cek poin saja untuk tim *development* pada metode *DevOps.*



Gambar 15. Perilisan aplikasi Kota Quest pada Github

Sebelum dirilis pada *Github*, folder aplikasi Kota Quest harus sudah dimasukkan ke dalam repositori *Github* terlebih dahulu. Caranya dengan melakukan. Langkah ini sudah dilakukan pada tahap *coding*, sehingga sudah tidak perlu melakukannya lagi (kecuali jika ada perubahan pada *Source Code* untuk diperbaharui). Aplikasi Kota Quest dirilis pada *Github* lengkap dengan *Source Code* dan *file* APK yang sudah siap diinstall pada perangkat *Android*. Aplikasi ini secara fundamental sudah dapat berjalan namun belum dapat digunakan oleh pengguna karena fitur penambahan *quest* membutuhkan aplikasi untuk di- *deploy* pada *Firebase.* Kota Quest App tersedia pada: https://github.com/ditocpp/Kota- Quest/releases/tag/1.0.

3.6. Peluncuran (Deploy)

Peluncuran aplikasi Kota Quest berbeda dengan perilisan aplikasi pada tahap kelima dari *DevOps* dimana pada perilisan bersifat mempublikasikan *build* aplikasi yang sudah dites dapat berjalan namun sebenarnya belum siap masuk dalam produksi dan penyebaran massal kepada pengguna. Lebih tepatnya *release* menandakan pembaharuan versi dari aplikasi yang belum disebarluaskan melalui tahap *deployment* ini. Seperti yang telah disebutkan pada tinjauan pustaka, *deployment* aplikasi *Quest Board* menggunakan *Firebase*. Implementasi dari teknologi *Firebase* yang digunakan pada aplikasi untuk melakukan *deployment* yaitu teknologi *Firebase Authentication* dan *Firestore Database*.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Menghubungkan Aplikasi dengan Firebase

Tahapan implementasi teknologi firebase yang pertama kali adalah menambahkan *Firebase service* (*google-services.json*) ke dalam aplikasi *Android*. *Firebase service* ini memuat informasi antarmuka untuk dapat menghubungkan aplikasi *Android* dengan *Firebase*.

Namun sebelumnya untuk mendapatkan *file* JSON ini perlu untuk memasukkan informasi aplikasi yang akan di-*deploy* pada *Firebase*, bisa dengan *Command Line* maupun *Console Firebase*. Untuk SHA-1 *certificate* bisa didapatkan melalui *CMD* dengan mengetikkan:

```
keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -aliasandroiddebugkey

-storepass android -keypass android
```

Maka akan muncul nomor seri SHA-1 yang dapat digunakan untuk delpoyment aplikasi ini.

Mengaktifkan Google Authentication

Teknologi *Firebase Authentication* ini dimanfaatkan pada proses login pengguna aplikasi dan pendaftaran calon pengguna aplikasi. Dalam hal ini dengan mengaktifkan *Enable* pada *Google Account Authentication* di *Firebase*. Sedangkan untuk realisasi pada aplikasi dengan menuliskan *Source code* di bawah ini pada *LoginActivity*:

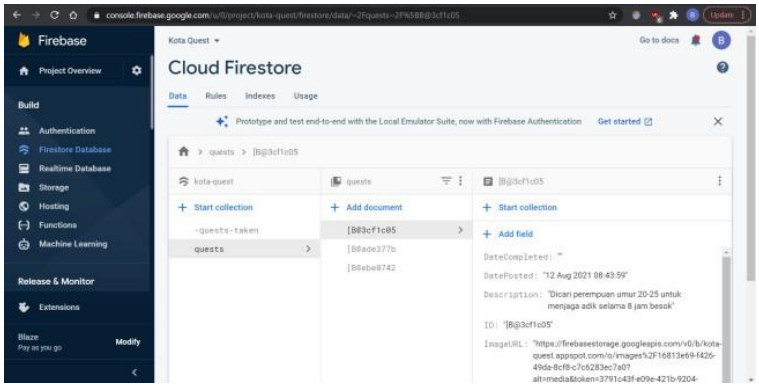
```
private fun firebaseAuthWithGoogle(idToken:String) {
    val credential = GoogleAuthProvider.getCredential(idToken, null)
    auth.signInWithCredential(credential)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UIwith the signed-in user's information
                Log.d("", "signInWithCredential:success")
                Log.d("", auth.currentUser?.uid.toString())
                uid = auth.currentUser?.uid.toString()
                Toast.makeText(this, "Signedin", Toast.LENGTH_LONG).show()

                val moveIntent = Intent(this,ActivityListQuests::class.java)
                startActivity(moveIntent)

            } else {
                // If sign in fails, display amessage to the user.
                Log.w("", "signInWithCredential:failure", task.exception)
                progressbar.visibility =
View.INVISIBLE
            }
        }
}
```

Menginisialisasi Firestore

Aplikasi Kota Quest membutuhkan tempat penyimpanan data informasi *quest* berbentuk JSON yang disimpan pada sebuah database. Dalam hal ini, database yang digunakan adalah *Firestore* yang ada pada *Firebase*. Untuk mengolah data yang akan disimpan pada server maka dibuatlah sebuah UI untuk menaruh data untuk dibaca dan diberikan oleh pengguna. *Cloud Firestore Database* dipilih yang servernya berada pada *asia-southeast2* atas dasar jarak yang dekat dengan tempat penelitian dilakukan.

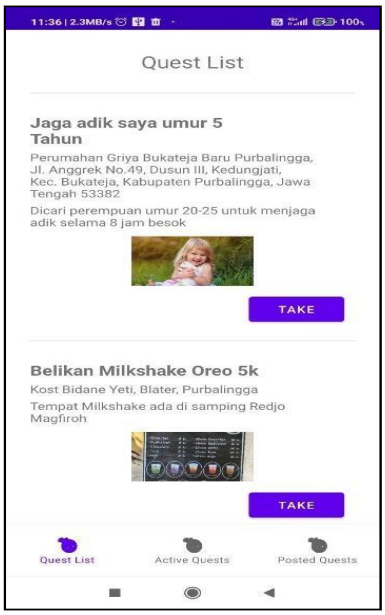


Gambar 16. Cloud Firestore Database.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

3.7. Pengoperasian (Operate)

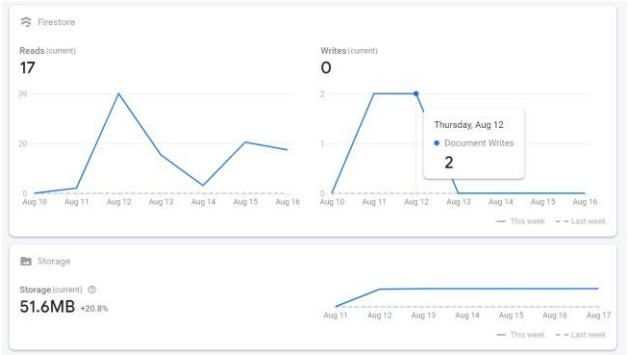
Pengoperasian aplikasi dilakukan untuk mengoperasikan aplikasi seolah-olah kita adalah penggunanya. Pada tahap penelitian ini ujicoba pengoperasian untuk aplikasi karena sebelumnya belum memungkinkan dikarenakan *Firebase* belum dipasang pada tahap *testing*. Pengoperasian aplikasi dimulai dengan membuka aplikasi, dimana akan muncul *splash screen* dengan logo Kota Quest. Kemudian muncul tampilan *login page* untuk melakukan *login* dengan *Google Account*. Pengguna kemudian langsung tertuju pada layar *Quest List*, yang berisi *quests* yang tersedia. Ada 3 menu di bawah yang dapat dipilih yaitu *Quest list*, *Active Quests*, dan *Posted Quests*



Gambar 17. *Quest List*

3.8. Pemantauan (Monitor)

Fase terakhir dari siklus *DevOps* adalah melakukan pemantauan terhadap aplikasi yang sudah disebarkan. Pada penelitian ini untuk memantau performa aplikasi dilakukan dengan *Analysis Cloud Firestore Database* yang ada pada *Firebase*. Dapat dilihat disini dari dua *quest* yang telah dibuat selama 1 minggu terakhir muncul sebagai 2 *Document Writes* pada *Firestore.* Sedangkan 17 *Document Reads* berarti ada 17 kali pengaksesan untuk melihat *Quests* pada aplikasi. *Storage* yang digunakan pada *Cloud Firestore Database* juga ada 51.6 *MegaBytes* yang berisi file JSON untuk menampung informasi setiap *quest* dan gambar yang telah di-*upload*. Dari sini dapat disimpulkan bahwa aplikasi sudah berjalan seperti yang diharapkan.



Gambar 18. Pemantauan Cloud Firestore Database.

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan maka diperoleh kesimpulannya sebagai berikut:

p-ISSN: 2775-4227
e-ISSN: 2775-4219

1. Pada umumnya, Metode *DevOps* lebih sering digunakan untuk sistem atau aplikasi yang berbasis web. Namun setelah melakukan penelitian ini, ternyata prinsip pendekatan *DevOps* dapat pula digunakan untuk aplikasi berbasis Android.
2. Metode *DevOps* memiliki 8 fase pengembangan yang merepresentasikan simbol tidak terhingga dengan artian bahwa suatu aplikasi atau sistem akan secara terus menerus melalui kedelapan fase tersebut selama aplikasi itu masih diperbaharui secara reguler.
3. *DevOps* bisa berarti lebih dari sekadar meningkatkan interaksi antara tim development dan operations saja. *DevOps* secara metodologi dapat menspesifikasikan informasi teknis mengenai apa saja yang dibutuhkan oleh sebuah aplikasi agar persyaratan sumber daya dapat terpenuhi melalui delapan fase yang sudah dijelaskan pada penelitian ini.
4. Penggunaan Firebase dapat memudahkan perancangan bagian back-end dari aplikasi yang dibuat, sehingga tidak perlu untuk membuat dan menyimpan data aplikasi menggunakan server sendiri.

Berdasarkan hasil penelitian, pengujian, implementasi serta pembahasan mengenai Rancang Bangun Aplikasi *Quest Board* untuk Masyarakat Menggunakan Metode *DevOps* Berbasis *Android*, maka untuk pengembangan penelitian selanjutnya penulis menyarankan sebagai berikut:

1. Mengembangkan aplikasi menjadi multi- platform yang dapat digunakan pada perangkat lainnya dengan sistem operasi yang berbeda seperti iOS.
2. Menambahkan Google Maps API pada input lokasi agar quest dapat terlacak secara lebih jelas.
3. Memberikan regression testing pada aplikasi di tahap testing atau monitoring untuk memastikan aplikasi benar-benar bisa berjalan lancar

DAFTAR PUSTAKA

[1] T. N. Effendi, "Budaya Gotong Royong Masyarakat Dalam Perubahan Sosial Saat Ini," *Jurnal Pemikiran Sosiologi*, vol. 2, no. 1, p. 1, Jan. 2016, doi: 10.22146/jps.v2i1.23403.

[2] S. Andrews, D. A. Ellis, H. Shaw, and L. Piwek, "Beyond Self-Report: Tools to Compare Estimated and Real-World Smartphone Use," *PLoS One*, vol. 10, no. 10, p. e0139004, Oct. 2015, doi: 10.1371/journal.pone.0139004.

[3] A. Taryana, A. Fadli, E. Murdyantoro, and S. Rahmah Nurshiami, "DevOps Approach Embraces Forward and Reverse Engineering," *International Journal of Applied Information Technology*, vol. 04, no. 02, 2020, doi: 10.25124/ijait.v4i02.2865.

[4] I. B. A. Y. Bharata, D. Mahariani, A. A. M. A. Dwiantari, K. S. Budiawan, N. N. T. Apriliyani, and F. Rahman, "PEMETAAN JALUR PENDAKIAN PADA KAWASAN HUTAN LINDUNG BUKIT CEMARA GESENG VIA DESA SILANGJANA MENGGUNAKAN APLIKASI GPS ALPHINE QUEST DAN GOOGLE EARTH PRO," *Jurnal ENMAP.*, vol. 2, no. 2, pp. 1–9, Sep. 2021, doi: 10.23887/em.v2i2.39131.

[5] A. Putra Pratama, A. Putra Kharisma, and R. C. Wihandika, "Pengembangan Aplikasi Daily Quest: Aplikasi Untuk Menangani Kemalasan Pada Anak Menggunakan Platform Android," 2019. [Online]. Available: http://j-ptiik.ub.ac.id

[6] S. Suparman, "MENEMUKAN KARAKTERISTIK BUTIR MENGGUNAKAN QUEST," *Al-Manar*, vol. 9, no. 1, pp. 83–104, Jun. 2020, doi: 10.36668/jal.v9i1.134.

[7] A. Taryana, A. Fadli, and S. R. Nurshiami, "Merancang Perangkat Lunak Sistem Penjaminan Mutu Internal (SPMI) Perguruan Tinggi yang Memiliki Daya Adaptasi Terhadap Perubahan Kebutuhan Pengguna secara Cepat dan Sering," *JURNAL Al-AZHAR INDONESIA SERI SAINS DAN TEKNOLOGI*, vol. 5, no. 3, pp. 121–129, Apr. 2020, doi: 10.36722/SST.V5I3.372.