JURNAL SAINS DAN INFORMATIKA
RESEARCH OF SCIENCE AND INFORMATIC V8.I2

Vol.08No.02(2022) 86-95 **p-issn : 2459-9549**
http://publikasi.lldikti10.id/index.php/jsi **e-issn : 2502-096X**

Implementasi *Platform As A Service* (PAAS) Pada Aplikasi *Getfix* Berbasis *Cloud Computing*

Sania Febriani[a], Fitri Purwaningtias[b]
[a]Sistem Informasi, Ilmu Komputer, Bina Darma, 191410221@student.binadarma.ac.id
[b] Sistem Informasi, Ilmu Komputer, Bina Darma, fitri.purwaningtias@binadarma.ac.id

Submitted: 19-10-2022, Reviewed: 07-11-2022, Accepted 22-11-2022
http://doi.org/10.22216/jsi.v8i2.1653

Abstract
Sometimes when the gadget used does not work because it is damaged, the user tries to make repairs. The problem is that users are often hampered from carrying out repairs due to a lack of information regarding the exact repair location or difficulty finding adequate technicians. From these problems, the author aims to design the Getfix application. The existence of Cloud Computing technology makes it easier to build this application, so the Google App Engine service based on Platform as A Service is applied which can automatically manage scalability and stable computing. In addition, Cloud Firestore is implemented as a NoSQL database for synchronizing stored data. The system development method uses the waterfall method and the system design uses the Unified Modeling Language. The final result of this research is an Android-based application that provides easier, faster, and more efficient cellphone, laptop, computer, and TV repair services.
Keywords: App Engine, Cloud Compuring, PaaS

Abstrak

Adakalanya ketika *gadget* yang digunakan tidak berfungsi karena mengalami kerusakan sehingga pengguna berusaha untuk melakukan perbaikan. Permasalahannya adalah pengguna sering kali terhambat melakukan perbaikan disebabkan kurangnya informasi mengenai lokasi perbaikan yang tepat atau sulit menemukan teknisi yang memadai. Dari masalah tersebut, penulis bertujuan untuk merancang aplikasi *Getfix*. Adanya teknologi *Cloud Computing* mempermudah membangun aplikasi ini, maka diterapkan layanan *Google App Engine* berbasis *Platform as A Service* yang secara otomatis dapat mengatur skalabilitas serta komputasi yang stabil. Selain itu diterapkan *Cloud Firestore* sebagai database NoSQL untuk sinkronisasi data yang tersimpan. Metode pengembangan sistem menggunakan metode *waterfall* dan perancangan sistem menggunakan *Unified Modeling Language*. Hasil akhir dari penelitian ini adalah sebuah aplikasi berbasis android yang menyediakan layanan perbaikan Handphone, Laptop, Komputer dan TV yang lebih mudah, cepat dan efisien.

Kata kunci: *App Engine, Cloud Computing, PaaS*

1. Pendahuluan

Getfix adalah aplikasi yang dirancang untuk memberikan informasi dan layanan jasa perbaikan *gadget* atau perangkat elektronik. Dengan semakin banyaknya pengguna *smartphone* membuat pengembangan aplikasi berbasis Android semakin banyak [1]. Berdasarkan data yang ditunjukkan oleh Kementrian Komunikasi dan Informatika menyatakan pada tahun 2018 terdapat lebih dari 100 juta orang Indonesia merupakan pengguna aktif *smartphone*. Hal itu pula yang membuat aplikasi ini dikembangkan menjadi aplikasi berbasis *Android* sehingga akan semakin mudah untuk diakses di masing-masing *smartphone.* Teknologi dan informasi begitu mudah untuk diakses saat ini, salah satu teknologi yang mengalami perkembangan saat ini adalah *cloud computing*. *Cloud Computing* adalah gabungan pemanfaatan teknologi (‘komputasi’) dan pengembangan berbasis *Internet* (‘awan’) [2]. Keberadaan teknologi *cloud computing* memungkinkan akses dari mana saja dan menggunakan perangkat *fixed* atau *mobile device* [3]. *Google App Engine (GAE)* merupakan layanan *Platform as a Service* (Paas). *PaaS* adalah model layanan *Cloud Computing* yang menyebarkan dan menjalankan perangkat lunak secara bebas sesuai keinginan, yang meliputi sistem operasi dan aplikasi. untuk memproses, menyimpan, ber-*internet*, maupun komputasi sumber daya lain yang penting [4]. *Cloud Computing* dengan karakteristiknya menawarkan skalabilitas penggunaan komputasi secara terdistribusi yang memberikan kecepatan komputasi

yang stabil [5]. Maka dari itu penelitian ini berfokus pada implementasi *Cloud Computing deployment Platform as a Service* (Paas) bernama *Google App Engine (GAE)* pada aplikasi *Getfix*. GAE menyediakan penskalaan otomatis dengan skalabilitas tinggi di mana biaya yang dibayarkan sesuai dengan apa yang digunakan. Tujuan dari penelitian ini adalah mengimplementasikan PaaS pada suatu aplikasi berbasis *android*, yang diharapkan dapat memberi manfaat dan menjadi solusi bagi pengguna yang membutuhkan informasi mengenai perbaikan *gadget* maupun perangkat elektronik yang mereka miliki serta mendatangkan pekerjaan bagi banyak teknisi.

2. Tinjauan Pustaka/Penelitian Sebelumnya

2.1 Kajian Pustaka

2.1.1 Android

Android adalah sistem operasi perangkat *mobile* yang semula dikembangkan oleh *Android Inc*. Perusahaan ini kemudian dibeli oleh *Google* pada tahun 2005. Android dirancang berdasarkan kernel Linux yang dimodifikasi [6]. Sistem operasi *Android* bersifat *open source* yang dapat digunakan dan dikembangkan secara *open source*, memungkinkan pengembang untuk mengelola, memodifikasi, atau bahkan membuat aplikasi mereka sendiri [7].

2.1.2 Mobile Application

Mobile Application adalah perangkat lunak berupa aplikasi yang dirancang menggunakan program komputerisasi untuk disematkan pada perangkat *mobile* seperti ponsel, tablet dan jam tangan digital [8].

2.1.3 Cloud Computing

C*loud Computing* bukan sebuah teknologi spesifik, melainkan sebuah model, menggambarkan model untuk penyediaan dan penggunaan infrastruktur *Information Technology* dan layanan serupa. *Cloud computing* menyediakan layanan penyimpanan terhadap data yang dimiliki oleh pengguna [9].

2.1.4 Google Cloud Platform

Layanan yang disediakan oleh *Google Cloud Platform* memungkinkan aplikasi untuk di-deploy dengan cepat. Artinya, aplikasi ini dapat diluncurkan dari sepotong kode sederhana yang hanya dapat dijalankan di *server* lokal ke *server* di *Internet*, yang dapat digunakan oleh banyak orang dalam waktu singkat.

2.1.5 Google App Engine (GAE)

Produk yang ditawarkan *Google Cloud Platform* terdiri dari *Google App Engine*, *Google Compute Engine*, *Google Cloud Storage, Google BigQuery, Google Cloud SQL, Google Prediction API* dan *Google Translation API. Google App Engine (GAE)* merupakan salah satu layanan *Platform as a Service (PaaS)* yang dimiliki oleh *Google Cloud Platform.*

Salesforce.com, *Heroku*, dan *Amazon Web Services* (*AWS*) memelopori teknologi PaaS pada tahun 2007. Setelah itu, pada tahun 2008, *Google* meluncurkan *App Engine* sebagai versi uji coba gratis. *PaaS* memungkinkan pengembangan aplikasi Internet dengan biaya rendah, efisien, dan skala besar [10].

Google App Engine (GAE) memungkinkan pengembangan aplikasi *web* dan *seluler* menggunakan berbagai bahasa runtime seperti *Python, Java, Node.js, PHP, Ruby, dan Go. Google App Engine* memiliki dua *environtment* yaitu standar dan *fleksibel*. Tipe standar dapat dengan cepat menskala dari nol hingga ribuan. Jenis *fleksibel* memerlukan setidaknya satu *instance* yang berjalan untuk setiap versi aktif dan dapat memakan waktu lebih lama untuk ditingkatkan sebagai *respons* terhadap lalu lintas. Tipe standar menggunakan algoritme penskalaan otomatis yang dirancang khusus.

2.1.6 Cloud Firestore

Cloud Firestore adalah layanan *Google Cloud Platform* untuk penyimpanan data *database NoSQL*. Berbeda dengan *database relasional*, di mana setiap item data memiliki hubungan dengan yang lain, *database NoSQL* ini berisi informasi yang tidak terkait langsung dengan yang lain [11]. Ketika data tersimpan di *Cloud Firestore*, maka data dapat diakses dimanapun dan dengan *stream*, data dapat diperbaharui secara *real-time*.

2.1.7 Application Program Interface (API)

Application Progamming Interface (API) adalah dokumentasi yang berisi *interface*, fungsi, kelas, struktur dan sebagainya [13]. Dengan menggunakan API, *developer* dapat mengembangkan atau mengintegrasikan dengan perangkat lunak lain. API memungkinkan aplikasi untuk saling terhubung. Bahasa pemrograman yang digunakan untuk membangung *BackEnd* pada penelitian ini adalah *Javascript*, *library NodeJs, Express* dan *Nodemon.*

2.1.8 Firebase Authentication

Mengetahui identitas pengguna memungkinkan aplikasi untuk menyimpan data pengguna dengan aman di *cloud* dan memberikan pengalaman pribadi yang sama di semua perangkat yang digunakan oleh pengguna tersebut [12]. *Firebase Authentication* membantu sistem autentikasi yang aman sekaligus meningkatkan proses masuk dan registrasi akun pengguna. Fungsi ini menawarkan solusi identitas *end-to-end*, termasuk akun email dan kata sandi, serta otentikasi untuk *Google*, *Twitter, Facebook, Github*, dan layanan lainnya. Ujicoba autentikasi yang dilakukan pada penelitian ini

menggunakan email, kata sandi dan login menggunakan akun *Google*.

Ketika klien melakukan registrasi akun dengan memasukan input *email, username, dan password* data akan dikirimkan ke database melalui *server App Engine NodeJS. Firebase Authentication* mengintegrasikan *Login* dengan *Google*, *Firebase* akan mengirim *token ID Google* pengguna yang akan ditukarkan dengan kredensial pengguna.

2.2 Penelitian Sebelumnya

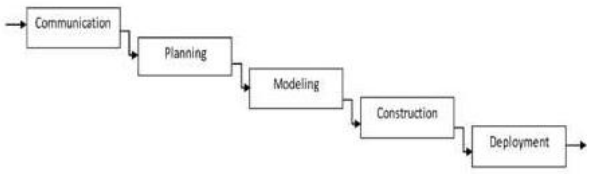
Penelitian yang dilakukan oleh Argo Wibowo adalah Perancangan Aplikasi Konsultasi Ibu Hamil Berbasis *Cloud Computing*. Penelitian ini menggunakan pendekatan berorientasi objek dengan model proses pengembangan *prototype* dan aplikasi yang dihasilkan memiliki 2 *server* yaitu *server* dari aplikasi dan *server cloud* milik *google Firebase* [13].

Pada penelitian sebelumnya yang dilakukan pengembangan aplikasi kemahasiswaan jurusan pendidikan teknik elektro (ASIK-JPTE) berbasis *cloud computing*. Penelitian ini menghasilkan aplikasi kemahasiswaan yang dibangun menggunakan Bahasa pemrograman *PHP, HTML, CSS,* dan *Javascript* [14].

Dari penelitian sebelumnya belum ada yang mengintegrasikan aplikasi dengan *Platform As A Service (PaaS)*. Maka dari itu, *server BackEnd Getfix* akan dijalankan menggunakan *Google App Engine*.

3. Metodologi Penelitian

Tahapan pengerjaan penelitian ini menggunakan *metode waterfall*. *Metode wartefall* adalah pendekatan sistematis dan berurutan yang dimulai dengan analisis sistem dan kebutuhan pengguna dan berkembang melalui tahap perencanaan seperti desain atau desain sistem dan basis data, pengkodean, pengujian, dan pemeliharaan sistem. [15].



Gambar 1. Metode Waterfall

3.1 Communication

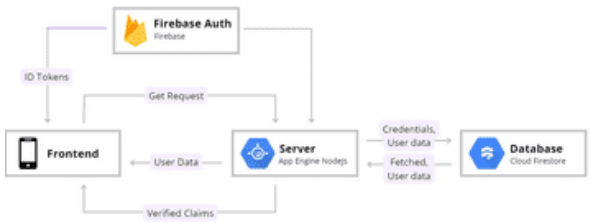
Tahapan ini terdiri dari proses identifikasi masalah dengan melakukan studi untuk mengetahui kebutuhan layanan sistem dari kendala, kapabilitas, teknologi melalui *brainstorming* dan konsultasi dengan mentor. Tahapan ini dilakukan secara langsung melalui virtual meeting. Setelahnya menentukan fungsional dan non-fungsional pada sistem. Hasil dari tahap ini berupa rancangan arsitektur *BackEnd* yang memanfaatkan teknologi *Cloud Computing* pada *Platform as a Service* (Paas) *Google Cloud Platform*.

3.2 Planning

Tahapan perencanaan ini terdiri dari proses *scoping, deliverables, scheduling, local deployment* dan *deployment Google Cloud Platform.*

3.3 Modeling

Tahap ini terdiri dari analisis dan desain sistem. Dilakukan analisa terhadap desain arsitektur pada layanan *Google Cloud Platform*, hasil dari proses ini adalah sebuah desain rancangan arsitektur pada layanan *Google Cloud Platform* yang dapat dilihat pada gambar 2.



Gambar 2. Arsitektur *Back-End* aplikasi *Getfix*

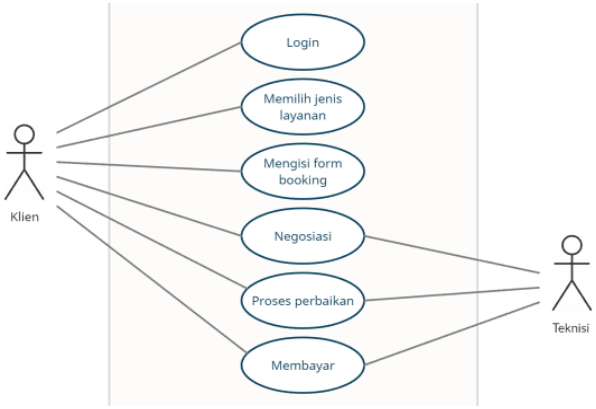
Pada gambar 2 menunjukkan perancangan arsitektur dari *Backend* aplikasi *Getfix*. Secara garis besar, *Frontend* akan mengirimkan *request* melalui *server App Engine Nodejs* kepada database Cloud Firestore. Data yang tersimpan di database akan di tampilkan ke frontend.

3.4 Construction

Tahapan ini terdiri dari penulisan kode program dan pengujian sistem. Tahapan penulisan kode program dilakukan dengan mengimplementasikan *RestFul API* menggunakan *Tools Visual Studio Code* dan testing kode program secara *local* dengan *Postman.Framework* yang digunakan pada *BackEnd* adalah *NodeJS, Express,* dan *Nodemon.*

a. *Use Case Diagram*

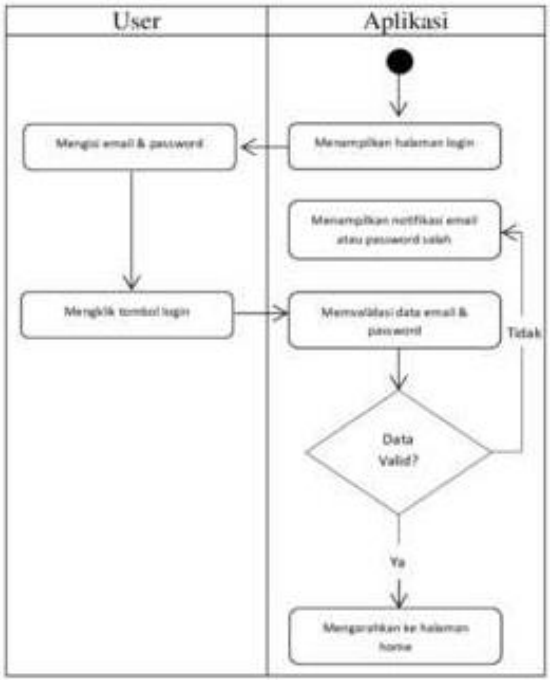
Pada gambar 3 menunjukkan *use case diagram* dari pengguna. Dimana pada gambar tersebut terdapat dua pengguna yaitu klien dan teknisi. Setelah klien login, klien bisa memilih jenis layanan pada halaman utama aplikasi, mengisi *formulir booking*, kemudian klien dan teknisi bisa bernegosiasi mengenai biaya perbaikan melalui fitur obrolan, jika terjadi kesepakatan teknisi akan menuju rumah klien untuk melakukan proses perbaikan.



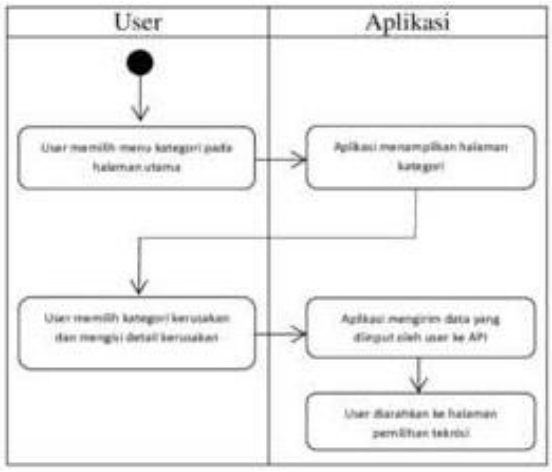
Gambar 3. *Use Case Getfix*

b. *Activity Diagram*

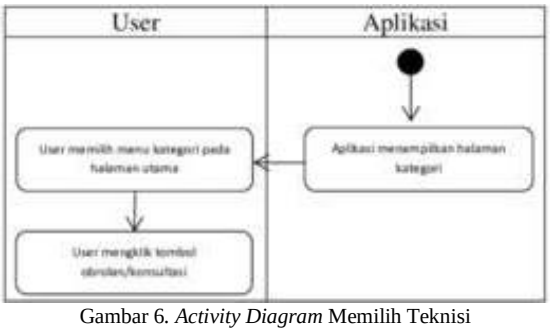
Activity Diagram digunakan untuk menggambarkan jalannya alur dalam aplikasi. Pada aplikasi *Getfix* terdapat aktivitas yang dideskripsikan dari pengguna aplikasi. Proses inti dari aplikasi *Getfix* setelah klien *Login* adalah memilih jenis layanan, mengisi *form booking* dan memilih teknisi.



Gambar 4. *Activity Diagram Login*



Gambar 5. *Activity* Diagram Memilih Jenis Layanan



Gambar 6. *Activity Diagram* Memilih Teknisi

3.5 Deployment

Pada tahap implementasi *Cloud Computing* dilakukan konfigurasi requirements yang di implementasikan pada *environment Google App Engine*. Selain itu dilakukan pemeliharaan sistem dengan memonitor trafik menggunakan fitur *Google Cloud Platform* yaitu *Log Explorer*.

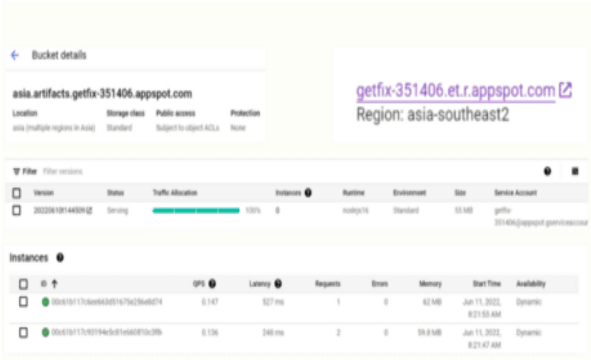
4. Hasil dan Pembahasan

Penelitian ini menghasilkan sebuah aplikasi bernama *Getfix* yang menyediakan informasi dan layanan reparasi. Pada bagian ini, akan dijabarkan tentang sistematika dari jalannya program aplikasi. Pelanggan terlebih dahulu melakukan registrasi akun pada aplikasi Getfix, kemudian pelanggan memilih kategori layanan pada menu utama. Aplikasi *Getfix* memiliki empat kategori layanan perbaikan yaitu pada *Handphone*, Laptop, Komputer dan Televisi. Setelah memilih kategori layanan, pelanggan mengisi *formulir booking* dan melengkapi data yang diperlukan seperti lokasi, tanggal, serta detail informasi yang diperlukan untuk menggunakan jasa panggil teknisi. Setelah itu sistem akan menampilkan rekomendasi teknisi yang sesuai dengan layanan yang dipilih pelanggan dan lokasi terdekat pelanggan sesuai *input* yang dilakukan pelanggan saat mengisi *formulir booking*. Lalu teknisi

akan mendatangi lokasi yang pelanggan berikan untuk melakukan pengecekan dan perbaikan.

4.1 Implementasi Cloud Computing

Pada gambar 7 menampilkan spesifikasi dari implementasi arsitektur *Getfix*. *Artifak deployment* aplikasi ini menggunakan *cloud storage* dengan spesifikasi *multiregion* di asia. Untuk *deployment Google App Engine*, digunakan *region asia-southeast1* (Jakarta) dengan *environment* standard dan *runtime nodejs16.*



Gambar 7. Spesifikasi arsitektur *Getfix*

4.2 Kode Program

```
//membuat orderan panggil teknisi
app.post('/orders',async (req,res)=>{
  const data = {
    id: nanoid(10),
        username : req.body.username,
        userTeknisi : req.body.userTeknisi,
    layanan: req.body.layanan,
    alamat: req.body.alamat,
    wilayah: req.body.wilayah,
    jadwal: req.body.jadwal,
        waktu : req.body.waktu,
        deskripsi: req.body.deskripsi,
        keterangan: "Pesanan Success"
  }
  await db.collection('orders').doc().set(data);
  res.json({message:"order success", data: {pesanan: data}});
})
```

Gambar 8. Kode Program Membuat Pesanan

Gambar 8 merupakan kode program untuk menampilkan data dari formulir pesanan. Data tersebut berupa layanan, alamat, wilayah, jadwal, waktu, deskripsi dan keterangan.

```
//menampilkan data orderan berdasarkan id
app.get("/orders/:id", async (req, res) => {
    const id = req.params.id;
    const query = db.collection('orders').where('id', '==', id);
    const querySnapshot = await query.get();
    if (querySnapshot.size > 0) {
        res.json(querySnapshot.docs[0].data());
    }
    else {
        res.json({status: 'Not found'});
    }
})
```

Gambar 9. Kode Program Menampilkan Informasi Pesanan Berdasarkan ID Pesanan

Gambar 9 merupakan kode progam untuk menampilkan informasi detail pesanan. Fungsi *app.get('/orders/:id'* melakukan *request* ke *database* untuk menampilakan informasi satu pesanan klien berdasarkan id pesanan.

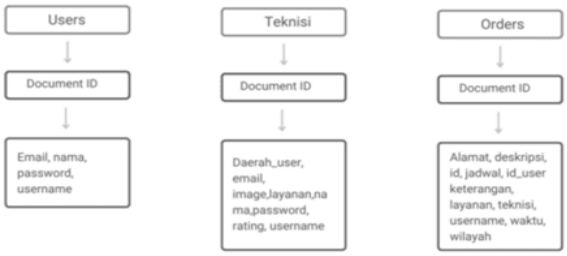
```
//menampilkan semua orderan user berdasarkan username
app.get('/allorder/:username', async (req, res) => {
  const username = req.params.username;
  let ord=[]
   const order = await db.collection('orders').where('username', '==', username).get()
  if (order.docs.length > 0) {
    for (const orders of order.docs) {
      ord.push(orders.data())
    }
  res.json(ord)
})
```

Gambar 10. Kode Program Menampilkan Pesanan Berdasarkan *Username*

Gambar 10 adalah kode program yang menampilkan semua pesanan yang dilakukan oleh seorang pengguna. Sistem *Getfix* dilengkapi fungsi yang menolak *username* yang sama, sehingga setiap *klien* memiliki *username* yang unik. Karena itu tidak akan terjadi kesalahan dalam menampilkan daftar pesanan.

4.3 Implementasi Basis Data

Dalam perancangan dan pembuatan database menggunakan *Cloud Firestore*. *Database* aplikasi memiliki tiga koleksi yaitu *Users*, Teknisi dan *Orders. Database* diagram dapat dilihat pada Gambar 11.



Gambar 11. Struktur *Database Getfix*

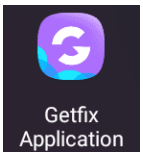
4.4 Tampilan Aplikasi

Berikut ini adalah hasil setelah tahap perancangan dari antarmuka aplikasi. Aplikasi Getfix memiliki *icon App Launcher*, halaman login, register, menu utama, formulir pemesanan, rekomendasi teknisi, detail

pesanan, riwayat pesanan, obrolan, profil dan halaman kamera.

a. Icon App Launcher

Ketika pelanggan mengunduh aplikasi Getfix pada ponsel, maka tampilan pada gambar 12 inilah yang akan pelanggan lihat. Desain Icon Aplikasi ini dibuat dengan aplikasi Ibis Paint X yang bisa di unduh dari Playstore atau App Store.



Gambar 12. Icon A*pp Launcher*

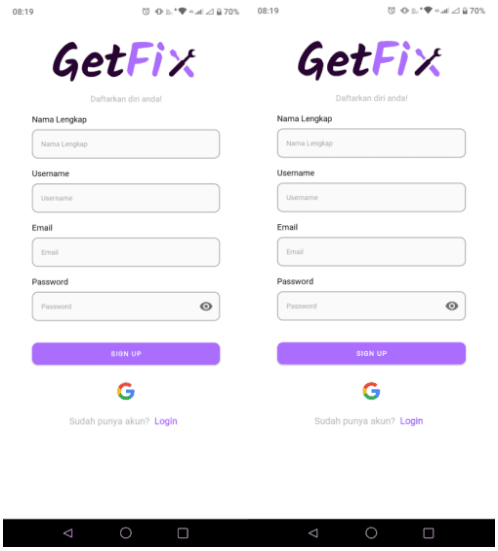
b. Splash Screen

Gambar 13. Splash Screen

Saat pelanggan pertamakali membuka aplikasi *Getfix*, halaman ini merupakan bagian pertama yang terlihat. Tampilan di gambar 13 adalah nama yang menjadi ciri khas Getfix.

c. Halaman Login dan Register

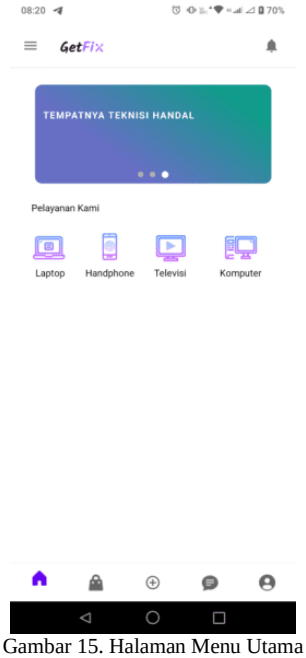
Pengguna baru dapat membuat akun baru dengan mengisikan data nama lengkap, *username*, *email* dan *password*. Pengguna juga bisa *login* dengan *google*, maka aplikasi ini akan melakukan autentikasi dengan layanan dari *firebase authentication*. Apabila proses autentifikasi benar maka pengguna dapat menggunakan fitur yang disediakan.



Gambar 14. Halaman Login dan Register

d. Halaman Menu Utama

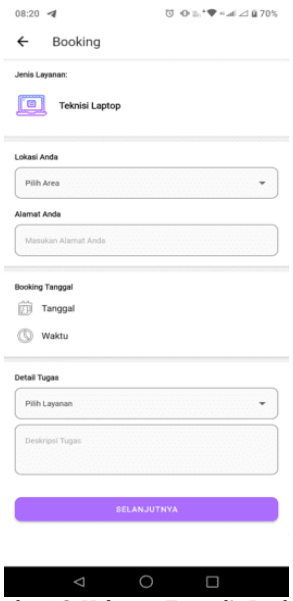
Pada menu utama terdapat beberapa komponen yang dapat digunakan oleh pelanggan, seperti: (1) Logo aplikasi yang terdapat pada bagian kiri atas aplikasi, (2) Icon notifikasi yang terdapat pada bagian kanan atas aplikasi, (3) Banner aplikasi yang menampilkan jargon dari aplikasi, (4) *Icon* jenis layanan yang terdiri dari laptop, *handphone*, televisi dan komputer, (5) Icon informasi pesanan. Berisi detail informasi pesanan dan riwayat pesanan, (6) *Icon* obrolan. Berisi percakapan antara klien dan teknisi, (7) Icon profil. Berisi detail informasi pengguna akun, (8) Icon + berfungsi untuk membuka halaman kamera.



Gambar 15. Halaman Menu Utama

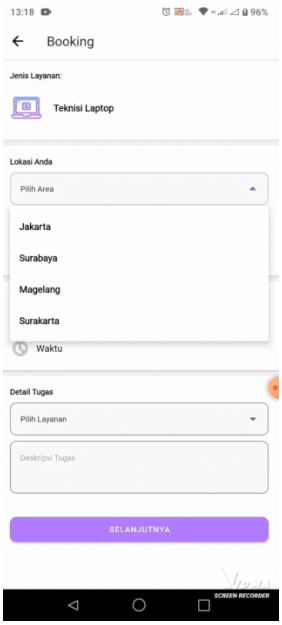
e. Halaman Formulir Booking

Halaman formulir pemesanan (*booking*) berisi tampilan data masukan yang tampil setelah pelanggan memilih jenis layanan pada menu utama. Data yang di simpan adalah jenis layanan, lokasi, alamat, tanggal, *detail* tugas, dan deskripsi bila diperlukan. *Formulir Booking* dapat dilihat pada Gambar 16.



Gambar 16. Halaman *Formulir Booking*

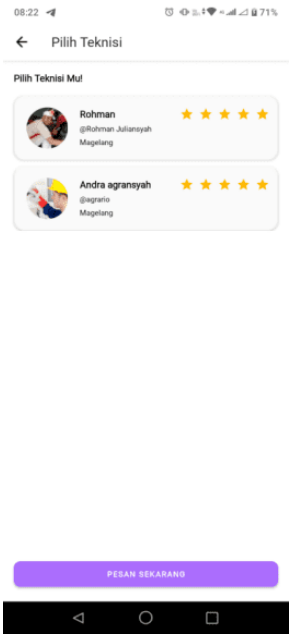
Saat ini *Getfix* diluncurkan sebagai *Minimum Variable Product (MVP)*. Karena itu jangkauan area pengguna hanya terdapat di empat wilayah yaitu: Jakarta, Surabaya, Magelang dan Surakarta. Pilihan layanan terdapat pada gambar 17.



Gambar 17. Area jangkauan *Getfix*

f. Halaman Rekomendasi Teknisi

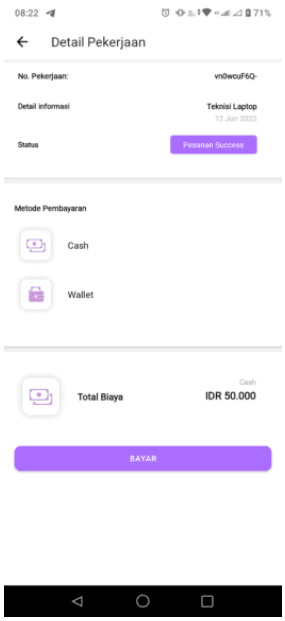
Halaman ini menampilkan beberapa teknisi yang direkomendasikan berdasarkan input data yang dimasukkan oleh pelanggan. Teknisi yang tampil adalah teknisi yang berada di daerah terdekat pelanggan beserta rating yang teknisi dapatkan.



Gambar 18. Halaman Rekomendasi Teknisi

g. Halaman Pembayaran Pesanan

Saat ini *Getfix* masih berupa *MVP* (*Minimum Variable Product*). Karena itu penulis mematok tarif sebesar Rp 50.000,- untuk wilayah yang ditentukan. Tarif ini bisa berubah sesuai kesepakatan antara pelanggan dan teknisi.



Gambar 19. Halaman Pembayaran Pesanan

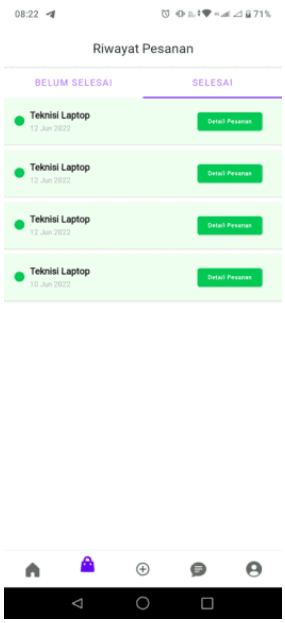
h. Halaman Detail Pesanan

Setelah mengisi formulir pemesanan jasa perbaikan maka aplikasi akan mengirim detail data pemesanan pada pelanggan. Pelanggan bisa menghubungi teknisi dengan fitur obrolan atau telepon. Jika kembali dari halaman ini, pelanggan akan ditujukan ke halaman menu utama.

Gambar 20. Halaman Detail Pesanan

i. Halaman Riwayat Pesanan

Gambar 21 menampilkan dua kategori riwayat pesanan, yaitu pesanan yang belum selesai dan pesanan selesai. Jika kembali dari halaman ini, pelanggan akan diarahkan ke halaman utama.



Gambar 21. Halaman Riwayat Pesanan

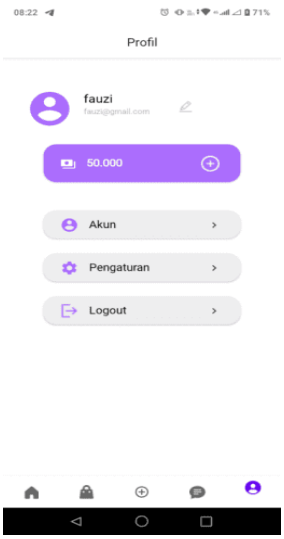
j. Halaman Obrolan

Pada halaman ini menampilkan ruang obrolan antara pelanggan dan teknisi, serta kolom dan halaman teks agar kedua pengguna dapat berinteraksi secara lansung.

Gambar 22. Halaman Obrolan

k. Halaman Profil

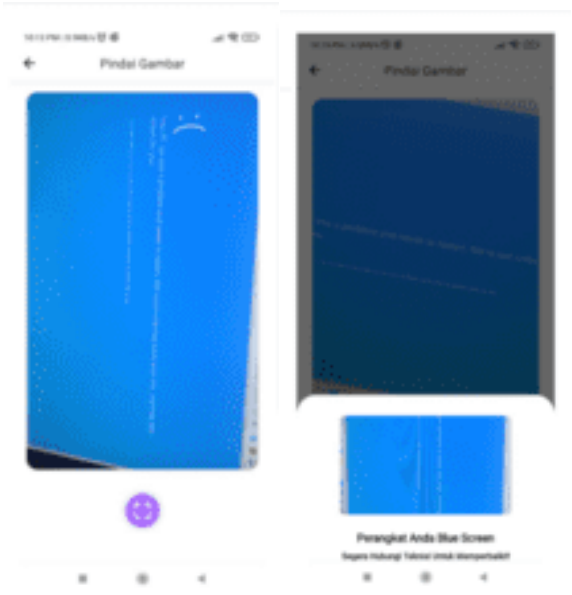
Kedua *user;* pelanggan dan teknisi memiliki halaman profil yang menampilkan data foto, nama, *email*, saldo, serta pengaturan akun dan tombol *logout*.



Gambar 23. Halaman Profil

l. Halaman Kamera/Prediksi Kerusakan

Tombol kamera/prediksi kerusakan terdapat pada halaman menu utama. Setelah pelanggan menekan tombol tersebut, pelangga akan diarahkan ke antarmuka seperti gambar 24. Saat ini klasifikasi dari model *machine learning* untuk prediksi kerusakan ada dua yaitu; layar *bluescreen* dan layar retak.



Gambar 24. Halaman Kamera/Prediksi Kerusakan

5. Kesimpulan

Implementasi *PaaS* pada aplikasi *Getfix* dilakukan dengan model pengembangan perangkat lunak W*aterfall* berhasil dibuat dan dapat digunakan kapan saja menggunakan koneksi *internet.* Aplikasi *Getfix* dapat berjalan baik pada sistem operasi *android* versi 7.0 – 9.0 (*Nougat* sampai *Pie*). Selain itu, fitur pendeteksi kerusakan berhasil menampilkan prediksi sesuai dengan kategori kerusakan.

Getfix dapat lebih optimal dengan penambahan kategori jenis layanan, klasifikasi pada pendeteksi kerusakan, dan juga *API Google Map* untuk menentukan tarif jarak yang ditempuh teknisi menuju lokasi klien seperti aplikasi transportasi online yang sukses saat ini. Selain itu, dapat juga menambah fitur pembayaran dengan *E-Money* agar *user* bisa melakukan pembayaran dan penarikan saldo melalui *virtual account*. Aplikasi untuk sisi teknisi juga akan membuat *Getfix* lebih berkembang agar teknisi bisa mengatur pesanan perbaikan yang masuk untuk diterima atau ditolak.

6. Daftar Rujukan

[1] N. K. C. Dewi, I. B. G. Anandita, K. J. Atmaja, And P. W. Aditama, "Rancang Bangun Aplikasi Mobile Siska Berbasis Android," P. 8.

[2] R. P. Sudirdja, "Pemanfaatan Teknologi Cloud Computing Dalam Reformasi Birokrasi Guna Mewujudkan Kejaksaan Yang Profesional, Komunikatif Dan Akuntabel," *J. Huk. Pembang.*, Vol. 50, No. 4, P. 828, May 2021, Doi: 10.21143/Jhp.Vol50.No4.2854.

[3] A. H. Jatmika, R. Afwani, And N. Agitha, "Perancangan Software As A Service (Saas) Untuk Sistem Pelayanan Kesehatan Ibu Dan Anak (Pkia) Pada Puskesmas Se-Kota Mataram Berbasis Cloud Computing," *J. Teknol. Inf. Dan Ilmu Komput.*, Vol. 6, No. 5, P. 485, Oct. 2019, Doi: 10.25126/Jtiik.2019651589.

[4] W. Setiawan, N. Fajriyah, And T. Duha, "Analisa Layanan Cloud Computing Di Era Digital," *J. Inform.*, Vol. 1, No. 1, P. 8, 2022.

[5] A. Arbain, M. A. Muhammad, T. Septiana, And H. D. Septama, "Learning Hoax News Pada Local Dan Cloud Computing Deployment Menggunakan Google App Engine," *J. Inform. Dan Tek. Elektro Terap.*, Vol. 10, No. 3, Aug. 2022, Doi: 10.23960/Jitet.V10i3.2646.

[6] N. Setyasmara, "Sistem Basis Data Pada Aplikasi Android Kalkulasi Biaya Cetak Buku," *J. Multi Media Dan It*, Vol. 5, No. 1, Aug. 2021, Doi: 10.46961/Jommit.V5i1.340.

[7] A. S. Handayani, S. Soim, A. F. S. Muhammad, N. Latifah, And R. Permatasari, "Aplikasi Air Detection Environment System (Adev) Dalam Mendeteksi Kadar Kualitas Udara Di Area Parkiran Berbasis Android," Vol. 7, No. 3, P. 16, 2020.

[8] "Aplikasi Absensi Mobile Berbasis Mapping Koordinat Lokasi (Studi Kasus : Lorus Celluler)," *J. Sains Dan Inform.*, Vol. 8, No. 1, Apr. 2022, Doi: 10.22216/Jsi.V8i1.893.

[9] T. Hidayat, "Encryption Security Sharing Data Cloud Computing By Using Aes Algorithm: A Systematic Review," Vol. 2, No. 2, P. 6, 2019.

[10] R. Yasrab, "Platform-As-A-Service (Paas): The Next Hype Of Cloud Computing," P. 21.
[11] E. D. Handoyo, S. Santoso, And D. J. Surjawan, "Pengembangan Aplikasi Mobile Pemesanan Dan Pembayaran Makanan Berbasis Cloud Storage," *J. Tek. Inform. Dan Sist. Inf.*, Vol. 8, No. 1, Apr. 2022, Doi: 10.28932/Jutisi.V8i1.4393.
[12] M. Ilhami, "Pengenalan Google Firebase Untuk Hybrid Mobile Apps Berbasis Cordova," *J. Ilm. It Cida*, Vol. 3, No. 1, Apr. 2018, Doi: 10.55635/Jic.V3i1.47.
[13] A. Wibowo, "Perancangan Aplikasi Konsultasi Ibu Hamil Berbasis Cloud Computing," *J. Matrik*, Vol. 17, No. 2, Pp. 68–79, May 2018, Doi: 10.30812/Matrik.V17i2.83.
[14] D. Vitalocca, E. S. Rahman, And N. M. Abdal, "Pengembangan Aplikasi Kemahasiswaan Jurusan Pendidikan Teknik Elektro (Asik-Jpte) Berbasis Cloud Computing," *J. Media Elektr.*, Vol. 18, No. 1, P. 1, Feb. 2021, Doi: 10.26858/Metrik.V18i1.19406.
[15] A. D. Samala And B. R. Fajri, "Rancang Bangun Aplikasi E-Sertifikat Berbasis Web Menggunakan Metode Pengembangan Waterfall," *J. Tek. Inform.*, Vol. 13, No. 2, Pp. 147–156, Feb. 2021, Doi: 10.15408/Jti.V13i2.16470.