
**TUGAS AKHIR - KI141502**

# Rancang Bangun Aplikasi Antrian *Online* Rumah Sakit Berbasis Android Menggunakan Parse sebagai *Backend as a Service* (BaaS)

KUKUH HANNUGROHO
NRP 5108100054

Dosen Pembimbing
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Isye Arieshanti, S.Kom, M.Phil.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

*[Halaman ini sengaja dikosongkan]*

**TUGAS AKHIR - KI141502**

# Rancang Bangun Aplikasi Antrian *Online* Rumah Sakit Berbasis Android Menggunakan Parse sebagai *Backend as a Service* (BaaS)

KUKUH HANNUGROHO
NRP 5108100054

Dosen Pembimbing
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Isye Arieshanti, S.Kom, M.Phil.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

*[Halaman ini sengaja dikosongkan]*


**FINAL PROJECT - KI141502**

# Design and Implementation of Online Hospital Queue Application on Android using Parse as Backend as a Service (BaaS)

KUKUH HANNUGROHO
NRP 5108100054

Advisor
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Isye Arieshanti, S.Kom, M.Phil.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

**Rancang Bangun Aplikasi Antrian *Online* Rumah Sakit Berbasis Android Menggunakan *Parse* sebagai *Backend as a Service* (Baas)**

## TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
**KUKUH HANNUGROHO**
NRP. 5108100054

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
   NIP: 198410162008121001 ..... (pembimbing 1)
2. Isye Arieshanti, S.Kom., M.Phil.
   NIP: 197804122006042001 ..... (pembimbing 2)

**SURABAYA**
**JUNI 2016**

*[Halaman ini sengaja dikosongkan]*

# RANCANG BANGUN APLIKASI ANTRIAN ONLINE RUMAH SAKIT BERBASIS ANDROID MENGGUNAKAN PARSE SEBAGAI BACKEND AS A SERVICE (BAAS)

**Nama Mahasiswa :Kukuh Hannugroho**
**NRP :5108100054**
**Jurusan :Teknik Informatika FTIf-ITS**
**Dosen Pembimbing I :Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
**Dosen Pembimbing II :Isye Arieshanti, S.Kom, M.Phil.**

## ABSTRAK

*Mengantri bagi sebagian orang adalah suatu aktifitas yang membosankan. Bahkan sebagian orang enggan untuk mengantri sehingga mereka keluar dari antrian sebelum mereka mendapatkan pelayanan. Bagi seorang pasien yang kondisi tubuhnya sedang tidak sehat, mengantri berjam-jam menambah ketidaknyamanan mereka. Waktu untuk mengantri akan jauh lebih efektif jika dipergunakan untuk kegiatan lainnya.*

*Pada tugas akhir ini akan dibuat sistem antrian online pada rumah sakit dengan menggunakan Parse sebagai Backend as a Service (BaaS). Sistem antrian online akan memudahkan pasien dalam melakukan antrian pada rumah sakit. Pasien dapat mengambil antrian dari rumah dan berangkat ke rumah sakit saat gilirannya telah mendekati.*

*BaaS merupakan model yang menjembatani pembuat aplikasi web dan mobile untuk menghubungkan aplikasi mereka dengan server. BaaS menyediakan fitur-fitur umum seperti manajemen pengguna, push notifications, dan integrasi dengan servis jaringan social. BaaS memudahkan developer dalam pengembangan aplikasi yang bersifat client-server. BaaS telah menyediakan infrastruktur utama pada server yang diperlukan oleh aplikasi secara umum.*

*Sistem ini diharapkan dapat membantu masyarakat dalam melakukan antrian di rumah sakit. Masyarakat dapat mengambil nomor antrian melalui sistem ini dari mana saja. Mereka dapat mempergunakan waktu menunggu antrian dengan kegiatan lain yang lebih bermanfaat.*

**Kata kunci: antrian, rumah sakit, parse, BaaS.**

**DESIGN AND IMPLEMENTATION OF ONLINE HOSPITAL QUEUE APPLICATION ON ANDROID USING PARSE AS BACKEND AS A SERVICE (BAAS)**


| | |
|---|---|
| **Name** | **:Kukuh Hannugroho** |
| **NRP** | **:5108100054** |
| **Jurusan** | **:Teknik Informatika FTIf-ITS** |
| **Advisor I** | **:Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.** |
| **Advisor II** | **:Isye Arieshanti, S.Kom, M.Phil.** |


## ABSTRACT


*Queue for some people is a boring activity. In fact, some people are reluctant to wait in line so that they come out of the queue before they get service. For a patient that his condition is not healthy, queue for hours adds to their discomfort. Time used to queue will be more effective if it is used for other activities.*

*In this final project will be made online queuing system at the hospital by using Parse as Backend as a Service (Baas). Online queuing system will make easier for patients to get queue for hospital services. Patients can take a queue from the house and went to the hospital when their turn approaching.*

*Baas is a model that bridges the web and mobile application developers to connect their applications to the cloud server. Baas provide common features such as user management, push notifications, and integration with social networking services. Baas will make easier for developers in the development of client-server applications. Baas has been providing the main infrastructure on the server side required by the application in general.*

*The system is expected to help people queuing at the hospital. They can take a queue number through the system from anywhere. They can use the time for waiting queue with other activities that are more useful.*



***Keywords: queue, hospital, parse, BaaS.***

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, kami panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada kami, sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang setinggi-tingginya kepada pihak-pihak yang telah membantu penulis dalam penyelesaian tugas akhir ini, terutama kepada:

1. Ibu Sri Wahyuni, orang tua penulis, yang menjadi motivasi penulis dalam menyelesaikan studi di Teknik Informatika ini dan yang selalu memberikan dukungan dan semangat agar penulis mampu untuk menyelesaikan tugas akhir, serta memberikan doa agar penyelesaian tugas akhir berjalan dengan lancar.
2. Agung Yulianto, Setyo Widanaryanti, dan Setyo Widiastuti, saudara-saudara penulis, yang selalu memberikan bantuan, semangat dan doa agar penulis mampu menyelesaikan tugas akhir.
3. Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. dan ibu Isye Arieshanti, S.Kom, M.Phil selaku dosen pembimbing yang telah memberikan banyak bantuan sehingga penulis dapat menyelesaikan tugas akhir ini.
4. Koordinator dan admin lab AJK yang telah memberikan tempat dan sarana sehingga penulis dapat mengerjakan tugas akhir hingga selesai.
5. Teman-teman Mahasiswa Teknik Informatika 2008 yang telah berjuang bersama-sama selama menempuh pendidikan di Jurusan ini.
6. Rekan-rekan grup “WG” yang telah banyak memberikan suntikan moral dan dorongan agar penulis tetap menyelesaikan studi di Informatika ini.

7. Serta pihak-pihak lain yang turut membantu penulis baik secara langsung maupun tidak, yang namanya tidak penulis sebutkan disini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, mohon maaf apabila ada kesalahan dan kata-kata yang dapat menyinggung perasaan. Penulis berharap tugas akhir ini dapat menjadi solusi bagi permasalahan antrian di rumah sakit.

Surabaya, Mei 2016

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

*[Halaman ini sengaja dikosongkan]*

# DAFTAR TABEL

*[Halaman ini sengaja dikosongkan]*

# BAB I
# PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

## 1.1. Latar Belakang

Mengantri bagi sebagian orang adalah suatu aktifitas yang membosankan. Bahkan sebagian orang enggan untuk mengantri sehingga mereka keluar dari antrian sebelum mereka mendapatkan pelayanan. Bagi seorang pasien yang kondisi tubuhnya sedang tidak sehat, mengantri berjam-jam menambah ketidaknyamanan mereka. Waktu yang dipergunakan untuk mengantri akan jauh lebih efektif jika dipergunakan untuk kegiatan lainnya.

Ada tiga tipe antrian yang digunakan dalam dunia nyata, yaitu antrian fisik, antrian virtual, dan antrian *mobile* [1]. Antrian fisik adalah antrian yang diwakilkan oleh pengantri itu sendiri dengan cara berdiri atau duduk berjajar. Antrian tipe ini sering digunakan dalam pembayaran di loket seperti supermarket, bioskop, dan taman bermain. Beberapa pelayanan kesehatan seperti dokter praktek masih menggunakan tipe antrian ini. Antrian fisik merupakan tipe antrian yang paling mudah dibuat karena tidak memerlukan sistem dan alat khusus.

Antrian virtual merupakan perkembangan dari antrian fisik dimana pengantri diwakilkan dengan benda lain, biasanya berupa kertas bernomor. Pengantri cukup mengambil nomor antrian dan menunggu di ruang tunggu. Antrian virtual bisa berupa manual ataupun komputerisasi. Antrian tipe ini dapat dijumpai di bank, stasiun kereta, dan pelayanan kesehatan seperti rumah sakit.

Kedua tipe antrian sebelumnya memiliki kelemahan yang sama yaitu pengantri harus datang ke lokasi dan menunggu

antriannya. Untuk mengatasi hal ini maka muncullah tipe antrian yang ketiga yaitu antrian mobile dimana pengantri bisa mengambil nomor antrian dari tempat lain tanpa perlu datang ke lokasi. Metode pengambilan nomor antrian dapat berupa melalui telepon dan internet. Antrian tipe ini masih jarang dijumpai di Indonesia. Padahal jumlah pengguna telepon pintar di Indonesia mengalami perkembangan yang sangat signifikan.

Berawal dari masalah ini penulis membuat aplikasi antrian *online* rumah sakit berbasis android. Penulis memilih android karena android memiliki jumlah pemakai yang paling besar. Aplikasi ini diharapkan dapat membantu pasien dalam mengambil nomor antrian dari mana saja dan memantau antrian yang ada. Pasien datang ke rumah sakit begitu nomor antrian sudah dekat sehingga pasien tidak perlu menunggu di rumah sakit terlalu lama. Aplikasi ini juga bermanfaat untuk meningkatkan kepuasan pasien terhadap pelayanan rumah sakit.

Dalam pengerjaan aplikasi ini, penulis menggunakan *Backend as a Service* (BaaS). BaaS menyediakan kerangka dan infrastruktur umum dari sisi *cloud server* yang dibutuhkan oleh aplikasi *mobile*. Hal ini dapat mempercepat proses pembangunan aplikasi karena developer tidak perlu membangun infrastruktur *server* yang rumit bagi aplikasi *mobile* yang dibangunnya.

## 1.2. Rumusan Permasalahan

Rumusan masalah yang ingin diselesaikan penulis dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana pengiriman dan pengambilan data antrian dari *server* menggunakan Parse?
2. Bagaimana menggunakan *push notification* untuk memberitahu pasien mengenai adanya perubahan antrian menggunakan Parse?

## 1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Aplikasi dibuat dalam bentuk aplikasi android dengan menggunakan bahasa pemrograman java.
2. Aplikasi dibuat dalam 2 tipe pengguna, yaitu pasien dan admin rumah sakit.
3. Aplikasi menggunakan Parse sebagai *Backend as a Service* (BaaS).
4. Push notification yang digunakan pada aplikasi adalah Google Cloud Messaging (GCM).
5. Aplikasi harus terhubung dengan internet.

## 1.4. Tujuan dan Manfaat

Tujuan dari pembuatan tugas akhir ini adalah:

1. Menyediakan aplikasi *mobile* yang dapat digunakan pasien untuk mengambil dan memantau antrian dari mana saja.
2. Meningkatkan efektivitas waktu pasien.
3. Meningkatkan kepuasan pasien terhadap pelayanan dokter.
4. Mempercepat proses development aplikasi dengan menggunakan BaaS.

*[Halaman ini sengaja dikosongkan]*

# BAB II
# DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini. Pokok permasalahan yang akan di bahas adalah tentang teknologi yang mendukung pembuatan tugas akhir.

## 2.1. Rancang Bangun Perangkat Lunak

Rancang bangun perangkat lunak merupakan tahap-tahap teknis untuk membangun perangkat lunak yang melingkupi analisis permasalahan dan kebutuhan, perencanaan, analisis sistem, implementasi, serta aktivitas pengujian dan pemeliharaan perangkat lunak.

Rancang bangun perangkat lunak diperlukan untuk menentukan konsep, strategi, dan praktik yang baik diterapkan untuk menciptakan perangkat lunak yang berkualitas tinggi, sesuai anggaran biaya, mudah dalam pemeliharaannya, serta tidak membutuhkan waktu yang lama dalam pembangunannya. Beberapa model rancang bangun perangkat lunak yang terkenal dan banyak dipakai antara lain: model air terjun dan model iterasi [2].

Model air terjun merupakan model yang berkembang secara sistematis dari satu tahap ke tahap lain. Model ini mengusulkan sebuah pendekatan kepada pengembangan software yang sistematik dan sekuensial yang mulai dari analisa kebutuhan, desain, implementasi, pengujian dan pemeliharaan. Model pengembangan ini bersifat *linear* dari tahap awal sampai tahap akhir pengembangan sistem. Tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan.

Model iterasi merupakan model yang berdasarkan pada ide pengembangan implementasi awal dan diujicobakan untuk mendapatkan tanggapan pengguna dan selanjutnya mengembangkannya beberapa versi hingga didapatkan sistem yang diinginkan. Model ini dimulai dengan menentukan dan

mengimplementasikan sebagian dari sistem yang akan dibuat dan diujicobakan untuk mendapatkan kebutuhan sistem berikutnya. Proses ini dilakukan beberapa kali dengan setiap proses menghasilkan versi baru dari aplikasi.

## 2.2. *Backend as a Service* (BaaS) dan Parse

BaaS merupakan cloud computing yang dibuat oleh suatu perusahaan untuk memudahkan developer dalam membangun dan mengoperasikan *cloud backend* untuk aplikasi mobile atau web mereka. BaaS menyediakan fitur-fitur umum yang diperlukan seperti manajemen pengguna, *push notification*, dan integrasi dengan *social networking services*. Fitur-fitur ini dapat diakses melalui *application programming interface* (API). Dengan adanya BaaS, developer tidak perlu membangun infrastruktur yang kompleks untuk sisi server aplikasi mereka. BaaS dapat membantu developer mempercepat proses pembuatan aplikasi [3].

Parse adalah salah satu layanan BaaS yang telah diakuisisi oleh Facebook. Parse menyediakan solusi backend yang lengkap untuk aplikasi *mobile*. Dengan menggunakan parse kita tidak perlu melakukan pengkodean pada server atau memelihara server. Dalam menggunakan parse, developer harus mendaftar pada website parse dan mendefinisikan aplikasi yang dibuat. Parse akan memberikan application id dan client key yang dibutuhkan dalam menggunakan parse pada aplikasi [4].

Parse telah menyediakan komponen-komponen yang diperlukan dalam membuat aplikasi mobile antara lain sebagai berikut.

### 2.2.1. Penyimpanan data

Parse menyediakan basis data untuk menampung data yang diperlukan oleh aplikasi mobile. Beberapa tabel data telah ditangani otomatis oleh Parse yaitu tabel *user* untuk memanajemen pengguna,

tabel *installation* untuk memanajemen instalasi aplikasi, dan tabel *session* untuk menangani sesi pengguna yang login ke dalam aplikasi.

### 2.2.2. Manajemen Pengguna

Parse menyediakan fitur untuk memanajemen pengguna seperti registrasi pengguna, login ke sistem, rekoveri kata sandi pengguna. Data pengguna dikelola pada tabel *user* pada basis data yang terletak di Parse *server*.

### 2.2.3. Push notification

Parse dapat mengirimkan notifikasi pada perangkat pengguna dengan memanfaatkan *Google Cloud Messaging* (GCM) untuk *smartphone* yang memiliki koneksi dengan *Google Play*. Bagi smartphone yang tidak memiliki koneksi ini, Parse menyediakan push notification melalui jaringan Parse sendiri.

### 2.2.4. Cloud Code

Parse menyediakan fitur yang dapat digunakan developer untuk melakukan pengkodean pada sisi server dengan menggunakan bahasa pemrograman Javascript. Fitur ini dapat dipakai untuk melakukan komputasi yang kompleks yang harus dijalankan pada server.

## 2.3. *Application Programming Interface* (API)

API merupakan sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh programmer saat membangun perangkat lunak untuk sistem operasi tertentu. Dalam Java, API dimasukkan ke dalam *package-package* yang sesuai dengan fungsinya [5].

## 2.4. *Push Notification*

*Push notification* merupakan pengiriman informasi dari suatu aplikasi *server* ke suatu perangkat *client* tanpa permintaan dari *client* tersebut. Hal ini berkebalikan dengan *pull notification* yang mengharuskan *client* untuk meminta informasi dari *server*. Keunggulan utama dari *push notification* dalam *mobile computing* adalah memungkinkan pesan untuk diterima meskipun aplikasi sedang tidak aktif [6].

## 2.5. *Google Cloud Messaging* (GCM)

*Google Cloud Messaging* (GCM) untuk Android adalah sebuah *service* yang mengijinkan adanya pengiriman data dari server ke pengguna Android yang *compatible*, dan menerima pesan dari perangkat yang memiliki koneksi yang sama dengan GCM *service* dengan melakukan antrian pesan dan dikirimkan ke target aplikasi Android yang sedang berjalan di perangkat target. GCM seutuhnya gratis sebesar apapun ukuran pesan yang akan dikirimkan maupun diterima dan tidak ada batasan kuota dalam penggunaannya. Untuk menggunakan GCM, setiap perangkat yang akan dilibatkan harus terdaftar dalam *web server* GCM. Hal ini berguna untuk menyimpan registrasi GCM ID masing-masing perangkat ke *webserver* dan satu perangkat hanya terdaftar satu registrasi GCM ID saja. Setiap aplikasi Android yang menggunakan GCM, di dalam kode sumber aplikasi tersebut telah dimasukkan fungsi GCM yang berisi inisialisasi registrasi GCM ID jika perangkat tersebut belum pernah didaftarkan sebelumnya, dan jika sudah terdaftar maka akan menyimpan nomor registrasi GCM ID perangkat tersebut menjadi session selama aplikasi tersebut berjalan. Tujuannya adalah untuk memanipulasi notifikasi pada tata kelola profil pengguna. Jika pengguna ingin mendapatkan notifikasi, maka dapat mengubah inisialisasi GCM notifikasi pengguna menjadi *true*, sedangkan jika pengguna tidak ingin mendapatkan notifikasi, maka dapat mengubah inisialisasi GCM notifikasi pengguna menjadi *false* [7].

# BAB III
# PERENCANAAN SISTEM

Bab ini membahas metodologi yang digunakan pada pengerjaan tugas akhir. Metodologi yang akan dijelaskan meliputi deskripsi umum dan arsitektur sistem yang merupakan bagian dari analisis dan perancangan sistem yang akan dibangun.

## 3.1. Deskripsi Sistem

Pada subbab ini akan dibahas mengenai penjelasan umum mengenai sistem dan model basis data yang dipakai dalam pengembangan sistem.

### 3.1.1. Deskripsi Umum Sistem

Pada tugas akhir ini akan dibangun sebuah system yang digunakan untuk mengelola antrian pada istansi kesehatan seperti rumah sakit atau puskesmas. Pasien dapat mengambil nomor antrian dari mana saja dengan menggunakan smartphone Android mereka. Pasien tidak perlu lagi mengantri terlalu lama di rumah sakit sehingga waktu yang dimiliki pasien lebih efektif.

Sistem akan dibangun dengan memanfaatkan teknologi BaaS yaitu Parse. Setiap permintaan dan penyimpanan data rumah sakit, poli, dan antrian dikirimkan ke Parse. Parse juga akan mengirimkan push notification untuk menginformasikan pasien yang telah tiba atau mendekati giliran antriannya sehingga pasien dapat memperkirakan waktu untuk datang ke rumah sakit yang bersangkutan.

### 3.1.2. Model Fisik Basis Data

Basis data disini teletak pada sisi server yaitu pada Parse. Semua permintaan yang berhubungan dengan basis data akan dikirim ke server parse. Setiap tabel yang dibuat di Parse

memiliki beberapa kolom default yang harus ada, yaitu *objectId*, ACL, *createdAt*, dan *updatedAt*. Parse memiliki beberapa tabel umum yang biasa dipakai untuk mengembangkan aplikasi mobile, yaitu tabel *User*, *Installation*, *Session*, *Role*, dan *Product*. 3 tabel akan dipakai dalam sistem ini yaitu *User*, *Installation* dan *Session*. Disamping itu ada 4 tabel tambahan yang berguna untuk menyimpan data rumah sakit dan antriannya.

#### 3.1.2.1. *User*

Tabel ini untuk menampung data pengguna aplikasi, dalam hal ini adalah pasien dan admin rumah sakit. Struktur tabel *user* dapat dilihat dilihat pada Tabel 3.1.

**Tabel 3.1 Struktur Table User**

| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek |
| username | String | Username dari pengguna, untuk mempermudah maka nilanya dibuat sama dengan email |
| name | String | Nama dari pengguna yang bersangkutan |
| email | String | Alamat surat elektronik dari pengguna |
| password | String | Kata kunci yang digunakan untuk masuk |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

#### 3.1.2.2. *Installation*

Tabel installation menyimpan data untuk setiap instalasi aplikasi di android. Struktur tabel *installation* dapat dilihat dilihat pada Tabel 3.2.

**Tabel 3.2 Struktur Tabel Installation**

| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek. |
| deviceToken | String | Token dari Google untuk mengirimkan *push notification* melalui GCM |
| user | Pointer <User> | Pengguna yang memiliki instalasi ini |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

### 3.1.2.3. *Session*

Tabel session menyimpan data login dari pengguna. Parse telah menangani sistem *login* pada aplikasi dan setiap sesi *login* itu akan disimpan di tabel ini. Struktur tabel *session* dapat dilihat dilihat pada Tabel 3.3.

**Tabel 3.3 Struktur Tabel Installation**

| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek. |
| sessionToken | String | Token dari sesi *login* |
| expiredAt | Date | Tanggal sesi akan kadaluarsa |
| user | Pointer <User> | Pengguna yang memiliki sesi ini |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

### 3.1.2.4. *Hospital*

Kelas ini menampung data rumah sakit. Struktur tabel *hospital* dapat dilihat dilihat pada Tabel 3.4.

**Tabel 3.4 Struktur Tabel Hospital**

| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek. |
| name | String | Nama rumah sakit |
| address | String | Alamat rumah sakit |
| picture | File | Foto rumah sakit |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

### 3.1.2.5. Poli

Kelas poli digunakan untuk menyimpan data poli dari setiap rumah sakit. Struktur tabel poli dapat dilihat dilihat pada Tabel 3.5.

**Tabel 3.5 Struktur Tabel Poli**

| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek. |
| hospital | Pointer <Hospital> | Rumah sakit tempat poli ini berada |
| name | String | Nama poli |
| queue Number | Number | Nomor antrian pada poli saat ini |
| queueTotal | Number | Total nomor antrian pada poli |
| open | Boolean | Status antrian buka ataukah tutup |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

### 3.1.2.6. *Queue*

Kelas queue digunakan untuk menyimpan antrian yang dilakukan pengguna pada suatu poli. Struktur tabel *queue* dapat dilihat dilihat pada Tabel 3.6.

**Tabel 3.6 Struktur Tabel Queue**

| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek. |
| poli | Pointer <Poli> | Antrian pada poli |
| createdBy | Pointer <User> | Pengguna yang membuat antrian ini |
| type | Number | Tipe dari antrian, 0 untuk pasien dan 1 untuk admin |
| number | Number | Nomor antrian |
| code | Number | Kode antrian yang dapat digunakan sebagai validasi saat dipanggil |
| status | Number | Status dari antrian, 1 untuk pasien datang memenuhi panggilan dan 2 jika tidak |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

### 3.1.2.7. *AdminHospital*

Kelas AdminHospital merupakan kelas penghubung antara kelas user dan kelas hospital. Kelas ini menyimpan data admin pada rumah sakit tertentu. Struktur tabel *AdminHospital* dapat dilihat dilihat pada Tabel 3.7.

**Tabel 3.7 Struktur Tabel AdminHospital**

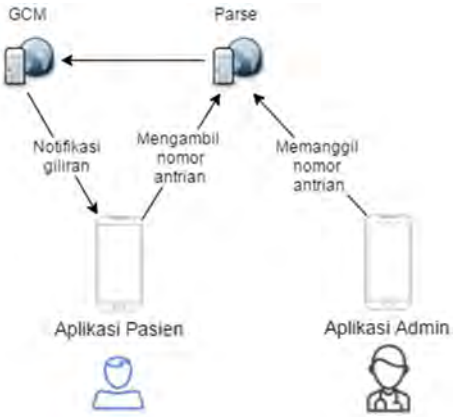
| Nama | Tipe Data | Deskripsi |
|---|---|---|
| objectId | String | ID obyek. |
| admin | Pointer <User> | Admin yang memiliki hak akses |
| hospital | Pointer <Hospital> | Rumah sakit yang bersangkutan |
| createdAt | Date | Tanggal dimana obyek ini dibuat |
| updatedAt | Date | Tanggal dimana obyek ini dilakukan perubahan |

## 3.2. Arsitektur Sistem

Subbab ini membahas rancangan sistem, meliputi: Arsitektur Umum Sistem dan Rancangan Proses Sistem.

### 3.2.1. Arsitektur Umum Sistem

Berdasarkan hasil analisa deskripsi umum sistem, secara garis besar arsitekur sistem dapat dibagi menjadi dua bagian, yakni pada *client* dan *server*. Modul *client* terdiri dari aplikasi android untuk pasien dan aplikasi android untuk admin rumah sakit. Sedangkan untuk *server* berupa Parse *server* dan Parse *Cloud Code*. Komunikasi antara client dan Parse *server* dilakukan melalui http dan *push notification*. *Push notification* untuk android memanfaatkan fasilitas *Google Cloud Messaging*. Arsitektur sistem secara keseluruhan terlihat pada Gambar 3.1




**Gambar 3.1 Arsitektur Sistem**

Pasien dapat melihat nomor antrian yang sedang berjalan pada suatu rumah sakit melalui aplikasi. Selanjutnya pasien dapat mengambil nomor antrian pada poli tertentu. Pengambilan nomor antrian ini dilakukan melalui HTTP request. Pasien dapat mengetahui berapa banyak antrian saat ini hingga sampai gilirannya nanti. Jika telah mendekati gilirannya, maka sistem akan mengirimkan notifikasi melalui GCM kepada pasien tersebut.

Admin rumah sakit dapat melihat berapa banyak total antrian yang ada dan nomor antrian saat itu. Selanjutnya admin dapat memanggil nomor antrian berikutnya. Pemanggilan nomor antrian juga dilakukan melalui HTTP request. Nomor antrian yang bersangkutan beserta beberapa nomor antrian selanjutnya akan mendapatkan notifikasi. Pasien dapat menunjukkan tampilan aplikasi sebagai bukti gilirannya. Untuk mengatasi permasalahan baterai habis saat gilirannya tiba, setiap nomor antrian disertai dengan 6 digit kode dan pasien dapat menunjukkan kode ini sebagai bukti gilirannya.

### 3.2.2. Rancangan Proses Sistem

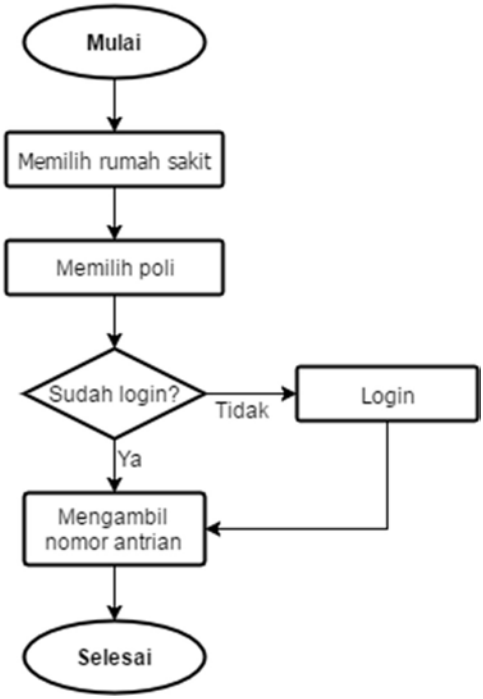
Pada rancangan proses sistem akan dijelaskan mengenai proses yang berjalan sistem untuk memenuhi fungsionalitas dengan bentuk diagram alur.

#### 3.2.2.1. Rancangan Proses pada Aplikasi Android untuk Pasien

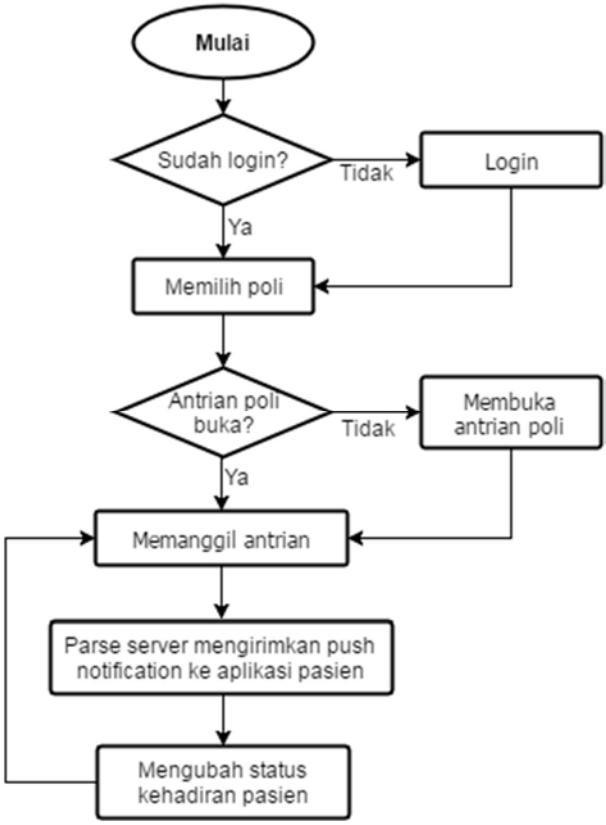
Proses pada aplikasi android untuk pasien dimulai dengan memilih rumah sakit yang dikehendaki. Sistem akan memberikan daftar poli yang ada pada rumah sakit tersebut beserta status antriannya. Pasien selanjutnya memilih poli dari daftar poli ini. Jika pasien belum *login* ke dalam sistem, maka pasien diharuskan untuk *login* terlebih dahulu. Selanjutnya jika pasien belum mengambil nomor antrian pada poli tersebut, maka pasien dapat mengambil nomor antrian. Rancangan proses ini dapat dilihat pada Gambar 3.2.

#### 3.2.2.2. Rancangan Proses pada Aplikasi Android untuk Admin Rumah Sakit

Proses pada aplikasi android untuk admin rumah sakit dimulai login ke dalam sistem. Sistem akan mencari rumah sakit dimana admin tersebut memiliki hak akses dan menampilkan daftar poli yang ada pada rumah sakit tersebut. Admin memilih salah satu poli. Bila antrian poli tersebut dalam keaadaan tutup, maka admin dapat membuka antrian sehingga pasien dapat mengambil nomor di poli tersebut. Admin selanjutnya bisa memanggil antrian dan mengubah status kehadiran pasien pada nomor tersebut. Bila mendekati waktu tutup poli, admin dapat mengubah status antrian menjadi *closed* sehingga pasien tidak dapat mengambil nomor antrian, sedangkan antrian yang tersisa masih dapat dipanggil. Rancangan proses ini dapat dilihat pada Gambar 3.3.




**Gambar 3.2 Diagram Alir Proses Pengambilan Nomor Antrian oleh Pasien**



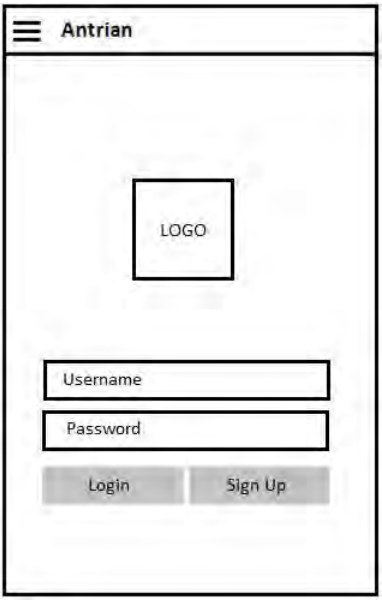

**Gambar 3.3 Diagram Alir Proses Pemanggilan Nomor Antrian oleh Admin**

### 3.2.3. Rancangan Antar Muka Aplikasi Android

Rancangan antarmuka aplikasi diperlukan untuk memberikan gambaran umum kepada pengguna bagaimana sistem yang ada dalam aplikasi ini berinteraksi dengan pengguna.

#### 3.2.3.1. Rancangan Antar Muka Login untuk Pasien dan Admin Rumah Sakit

Antar muka login untuk pasien dan admin memiliki rancangan yang sama. Hanya saja admin diharuskan untuk melakukan login saat awal masuk ke aplikasi. Hal ini berbeda dengan pasien. Pasien dapat melihat daftar rumah sakit dan daftar poli tanpa perlu login ke aplikasi. Pasien baru diharuskan login bila ingin mengambil nomor antrian. Rancangan antar muka login terlihat pada Gambar 3.4.

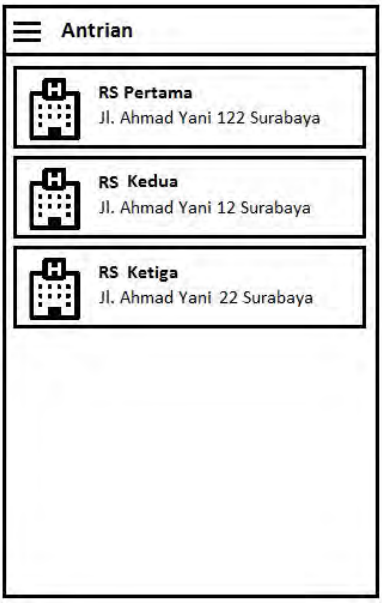



**Gambar 3.4 Rancangan Antar Muka Login**

#### 3.2.3.2. Rancangan Antar Muka Daftar Rumah Sakit yang Tersedia untuk Pasien

Pada saat aplikasi dibuka, pasien akan diberikan tampilan daftar rumah sakit yang tersedia. Data rumah sakit yang

ditampilkan berupa nama rumah sakit, alamat, dan gambar dari rumah sakit tersebut. Rancangan antar muka daftar rumah sakit terlihat pada Gambar 3.5.



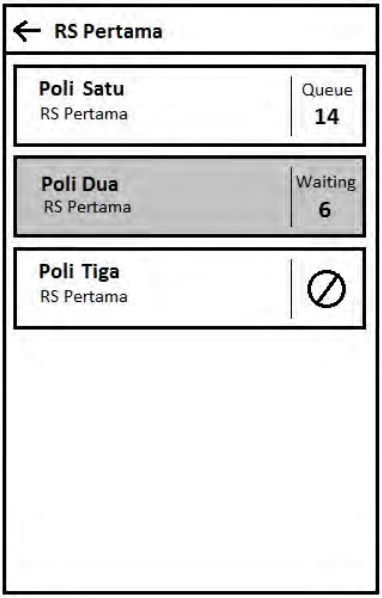

**Gambar 3.5 Rancangan Antar Muka Daftar Rumah Sakit**

### 3.2.3.3. Rancangan Antar Muka Daftar Poli pada Rumah Sakit untuk Pasien

Dengan menekan salah satu rumah sakit pada tampilan daftar rumah sakit, aplikasi selanjutnya akan menampilkan daftar poli yang ada pada rumah sakit tersebut. Informasi yang disertakan di setiap poli adalah nama poli dan status antrian di poli tersebut. Status antrian yang dimaksud meliputi :

- Antrian tutup, akan ditampilkan tanda tutup
- Antrian buka namun pasien belum mengambil nomor antrian, maka akan ditampilkan seberapa banyak antrian yang sedang berjalan di poli tersebut.

- Pasien telah mengambil nomor antrian, maka akan ditampilkan berapa antrian lagi hingga tiba gilirannya.




**Gambar 3.6 Rancangan Antar Muka Daftar Poli pada Rumah Sakit**

### 3.2.3.4. Rancangan Antar Muka Antrian pada Poli untuk Pasien

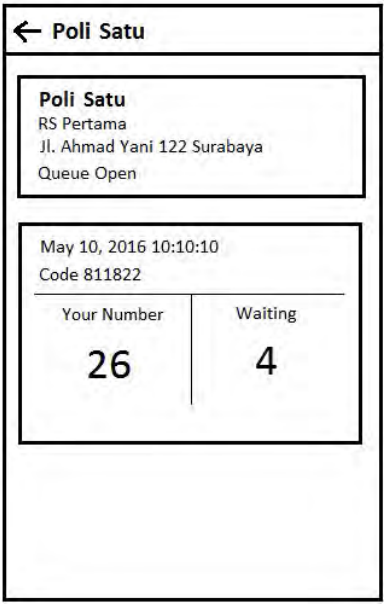
Antar muka antrian pada poli dapat diakses jika poli tersebut dalam keadaan buka. Pasien juga harus dalam kondisi login ke dalam aplikasi. Jika belum login maka aplikasi akan menampilkan halaman login.

Antar muka antrian pada poli ada dua macam, yaitu antar muka ketika pasien belum mengambil nomor antrian dan antar muka ketika pasien telah mengambil nomor antrian. Bagi pasien yang belum mengambil nomor antrian, aplikasi akan menampilkan nomor antrian saat itu dan total antrian pada poli yang bersangkutan. Rancangan antar muka antrian pada poli bagi

pasien yang belum mengambil nomor antrian terlihat pada Gambar 3.7.


**Gambar 3.7 Rancangan Antar Muka Poli Sebelum Mengambil Nomor Antrian**

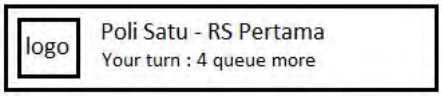
Bagi pasien yang sudah mengambil nomor antrian, aplikasi akan menampilkan detail nomor antrian yang telah diambil, meliputi nomor antrian pasien, tanggal pengambilan nomor, kode antrian, dan berapa giliran lagi hingga nomornya tersebut. Rancangan antar muka antrian pada poli bagi pasien yang telah mengambil nomor antrian terlihat pada Gambar 3.8.




**Gambar 3.8 Rancangan Antar Muka Poli Setelah Mengambil Nomor Antrian**

### 3.2.3.5. Rancangan Antar Muka Notifikasi Giliran Antrian Telah Tiba / Mendekati

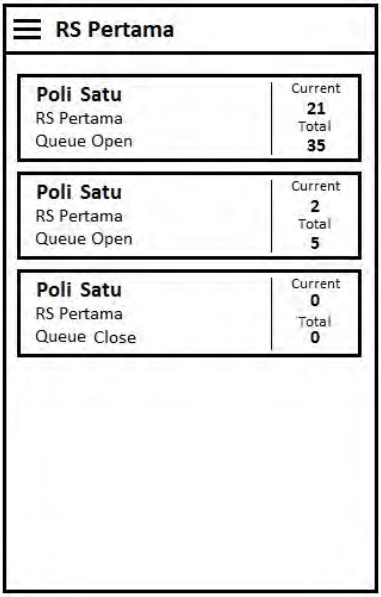
Ketika admin memanggil nomor antrian pada suatu poli, sistem akan mengirimkan notifikasi melalui GCM ke pengguna dengan nomor antrian yang dipanggil dan 5 nomor antrian selanjutnya. Pada perangkat pengguna, notifikasi yang ditampilkan adalah data poli, rumah sakit, dan berapa giliran lagi hingga nomor antriannya dipanggil. Rancangan antar muka dari notifikasi giliran antrian tersebut terlihat pada Gambar 3.9.




**Gambar 3.9 Rancangan Antar Muka Notifikasi Giliran**

### 3.2.3.6. Rancangan Antar Muka Daftar Poli pada Rumah Sakit untuk Admin Rumah Sakit

Pada dasarnya 1 admin bisa memiliki relasi dengan beberapa rumah sakit namun pada kenyataannya mayoritas seseorang hanya bekerja pada 1 rumah sakit saja. Untuk itu halaman awal antar muka aplikasi bagi admin adalah daftar poli pada rumah sakit dimana admin tersebut bekerja. Antar muka daftar poli ini mirip dengan antar muka pada aplikasi pasien, namun dengan perbedaan data yang ditampilkan pada tiap poli adalah nomor antrian saat ini dan total nomor antriannya. Rancangan antar muka tersebut terlihat pada Gambar 3.10.

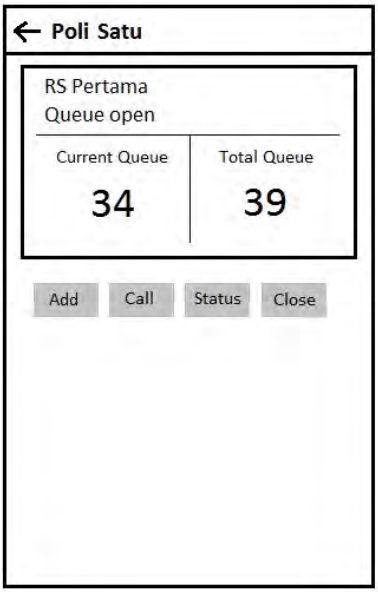



**Gambar 3.10 Rancangan Antar Muka Daftar Poli untuk Aplikasi Admin**

### 3.2.3.7. Rancangan Antar Muka Antrian pada Poli untuk Admin Rumah Sakit

Di bagian ini admin dapat mengatur antrian poli. Ada 4 tombol untuk mengatur antrian yaitu :

- Tombol *open/close* : Tombol untuk mengatur kondisi antrian buka ataukah tutup.
- Tombol *add* : Tombol untuk mengambil nomor antrian. Tombol ini berguna untuk pasien yang mengambil nomor antrian langsung di rumah sakit.
- Tombol *call* : Tombol untuk memanggil nomor antrian selanjutnya.
- Tombol status : Tombol untuk mengatur status kedatangan nomor antrian yang sedang dipanggil.




**Gambar 3.11 Rancangan Antar Muka Poli untuk Aplikasi Admin**

*[Halaman ini sengaja dikosongkan]*

# BAB IV
# IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem yang telah dijabarkan pada bab sebelumnya.

## 4.1. Lingkungan Pembangunan

Dalam membangun sistem ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

### 4.1.1. Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem ini adalah sebagai berikut:

- 1 PC dengan prosesor Intel Core i3 CPU @ 3,50GHz dan memori (RAM) 4,00 GB.
- 1 *Smartphone* Android dengan sistem operasi Marshmallow 6.1.

### 4.1.2. Lingkungan Pembangunan Perangkat Lunak

- Windows 7.0 32 bit sebagai sistem operasi.
- Android Studio versi 2.1 untuk melakukan pemrograman aplikasi Android.
- Google Chrome versi 50.0.2661.102 untuk mengakses web aplikasi Parse.
- Parse command line tool for Windows untuk melakukan pemrograman Parse *Cloud Code*.
- Notepad++ versi 6.9.1 untuk melakukan pemrograman javascript yang digunakan pada Parse *Cloud Code*.

## 4.2. Implementasi Sistem

Pada sistem ini perancangan sistem diimplementasikan ke dalam beberapa bagian yaitu aplikasi Android bagi pasien, aplikasi Android bagi admin rumah sakit, dan implementasi pada Parse.

### 4.2.1. Implementasi pada Parse *Cloud Code*

*Cloud code* digunakan untuk melakukan komputasi yang kompleks sehingga pengaksesan ke Parse menjadi lebih cepat. Pada sistem antrian online ini terdapat 5 fungsi kompleks yang dilakukan di *cloud code*.

#### 4.2.1.1. Fungsi Membuka Antrian Poli

Fungsi ini digunakan oleh admin rumah sakit untuk membuka antrian sehingga pasien dapat mengambil antrian pada poli tertentu. Fungsi ini menerima parameter berupa id poli. Bila poli yang bersangkutan dalam status *closed*, maka fungsi ini akan mengubahnya menjadi *open* dan mereset nilai dari *queueNumber* dan *queueTotal*. Kode sumber fungsi terdapat pada Gambar 4.1.

#### 4.2.1.2. Fungsi Menutup Antrian Poli

Fungsi ini digunakan oleh admin rumah sakit untuk menutup antrian poli sehingga pasien tidak dapat lagi mengambil nomor antrian pada poli tersebut. Fungsi ini menerima parameter berupa id poli. Bila poli yang bersangkutan dalam status *open*, maka fungsi ini akan mengubahnya menjadi *closed*. Kode sumber fungsi terdapat pada Gambar 4.2.

```
Parse.Cloud.define('openQueue', function(request, response) {
    findPoli(request, response, function(poli) {
        if (poli.get('open')){
            response.error('Already open');
            return;
        }

        poli.set('queueNumber', 0);
        poli.set('queueTotal', 0);
        poli.set('open', true);
        poli.save(null, {
            success: function(poli){
                response.success({poli: poli});
            }, error: function(detail, error) {
                response.error(error.message);
            }
        });
    });
});
```

**Gambar 4.1 Kode Sumber Fungsi Membuka Antrian Poli**

```
Parse.Cloud.define('closeQueue', function(request, response) {
    findPoli(request, response, function(poli) {
        if (!poli.get('open')){
            response.error('Already closed');
            return;
        }

        poli.set('open', false);
        poli.save(null, {
            success: function(poli) {
                response.success({poli: poli});
            }, error: function(detail, error) {
                response.error(error.message);
            }
        });
    });
});
```

**Gambar 4.2 Kode Sumber Fungsi Menutup Antrian Poli**

#### 4.2.1.3. Fungsi Menambah Antrian Poli

Fungsi ini digunakan baik oleh admin rumah sakit maupun pasien untuk mengambil nomor antrian poli. Fungsi ini menerima parameter berupa id poli. Pengambilan nomor dapat dilakukan jika poli yang bersangkutan dalam status *open*. Selanjutnya obyek queue akan dibuat dengan tipe sesuai penggunanya, 0 untuk pasien dan 1 untuk admin. Queue memiliki kode yang dapat digunakan untuk memvalidasi pasien seandainya smartphone yang dimilikinya sedang mati. Kode sumber fungsi terdapat pada Gambar 4.3.

```
Parse.Cloud.define('addQueue', function(request, response) {
    findPoli(request, response, function(poli) {
        var number = poli.get('queueTotal') + 1;
        var queue = new Parse.Object('Queue');

        poli.set('queueTotal', number);
        poli.save();
        queue.save({
            poli: poli,
            createdBy: Parse.User.current(),
            type: request.params.type ? 1 : 0,
            number: number,
            status: 0,
            code: Math.floor(Math.random() *
                (999999 - 100000 + 1)) + 100000
        },{
            success: function(queue) {
                response.success({
                    poli: poli,
                    newQueue: queue
                });
            }, error: function(queue, error) {
                response.error(error.message);
            }
        });
    });
});
```

**Gambar 4.3 Kode Sumber Fungsi Menambah Antrian Poli**

#### 4.2.1.4. Fungsi Memanggil Antrian Poli

Fungsi ini digunakan oleh admin rumah sakit untuk memanggil nomor antrian berikutnya. Fungsi ini menerima parameter berupa id poli. Infromasi antrian akan dikirim dengan menggunakan *push notification* kepada 5 nomor antrian berikutnya. Kode sumber fungsi terdapat pada Gambar 4.4.

#### 4.2.1.5. Fungsi Mengubah Status Kehadiran Nomor Antrian

Fungsi yang kelima adalah mengubah status antrian. Fungsi ini digunakan oleh admin rumah sakit untuk mengubah status apakah pasien dengan nomor antrian yang dipanggil hadir ataukah tidak. Kode sumber fungsi terdapat pada Gambar 4.5.

### 4.2.2. Implementasi Aplikasi Android

Implementasi aplikasi Android baik bagi pasien maupun bagi admin rumah sakit dilakukan dalam satu project android. Hal ini dapat dilakukan dengan menggunakan *product flavors*. *Product flavors* mendefinisikan versi kustom dari suatu aplikasi yang dibuat pada suatu project. Satu project dapat mempunyai beberapa *flavors* yang akan menghasilkan aplikasi yang berbeda.

Dengan menggunakan product flavors ini maka beberapa resource yang sama tidak perlu membuat dua kali, seperti *resource* gambar, beberapa kelas umum, *layout*, dsb. *Product flavors* didefinisikan di dalam build.gradle, seperti yang terlihat pada Gambar 4.6.

```
Parse.Cloud.define('callQueue', function(request, response) {
    var user = Parse.User.current();
    findPoli(request, response, function(poli) {
        var number = poli.get('queueNumber');
        var total = poli.get('queueTotal');
        if (number >= total) {
            response.error('No queue available');
            return;
        }
        number++;
        var diff = (total - number >= 4) ? 4 : total - number;
        for (var i = 0; i <= diff; i++) {
            var qQuery = new Parse.Query('Queue');
            qQuery.equalTo('poli', poli);
            qQuery.equalTo('number', number + i);
            qQuery.equalTo('status', 0);
            qQuery.equalTo('type', 0);

            var iQuery = new Parse.Query(Parse.Installation);
            iQuery.equalTo('appName', 'Antrian');
            iQuery.matchesKeyInQuery('user', 'createdBy', qQuery);

            Parse.Push.send({
                where: iQuery,
                data: {
                    hospitalName: poli.get('hospital').get('name'),
                    poliName: poli.get('name'),
                    turn: i
                }
            });
        }

        poli.increment('queueNumber');
        poli.save();
        var query = new Parse.Query('Queue');
        query.equalTo('poli', poli);
        query.equalTo('number', number);
        query.first({
            success: function(queue){
                response.success({poli: poli, queue: queue});
            }, error: function(object, error) {
                response.error(error.message);
            }
        });
    });
});
```

**Gambar 4.4 Kode Sumber Fungsi Memanggil Antrian Poli**

```
Parse.Cloud.define('statusQueue', function(request, response) {
    var query = new Parse.Query('Queue');
    query.get(request.params.queue, {
        success: function(queue) {
            queue.set('status', request.params.status);
            queue.save(null, {
                success: function(queue) {
                    response.success(queue);
                }, error: function(queue, error) {
                    response.error(error.message);
                }
            });
        }, error: function(object, error) {
            response.error(error.message);
        }
    });
});
```

**Gambar 4.5 Kode Sumber Fungsi Mengubah Status Nomor Antrian**

```
productFlavors {
    user {
        applicationId "com.okasoft.antrian.user"
        versionName "1.0-user"
    }
    admin {
        applicationId "com.okasoft.antrian.admin"
        versionName "1.0-admin"
    }
}
```

**Gambar 4.6 Kode Sumber Product Flavors**

Hal pertama dalam menggunakan Parse adalah mendaftar pada website Parsi yakni parse.com. Selanjutnya kita dapat membuat aplikasi baru untuk server antrian pada website parse dan akan mendapat *application ID* dan *client key*. Kedua nilai ini diperlukan dalam penggunaan parse pada aplikasi android. Penggunaan Parse SDK pada aplikasi android harus menginisiasi Parse terlebih dahulu di fungsi *onCreate* pada kelas *Application* dengan memanggil fungsi *initialize* dengan menyertakan

parameter *applicationId* dan *clientKey*. Inisiasi Parse dapat dilihat pada Gambar 4.7.

Pada aplikasi antrian ini terdapat 3 model data yang digunakan yaitu *Hospital*, *Poli*, dan *Queue*. Ketiga model ini merupakan extends dari *ParseObject*. *ParseObject* merupakan model yang digunakan untuk pendefinisian setiap kelas data pada Parse. Pengaksesan nilai kolom tabel pada *ParseObject* dilakukan dengan fungsi *set* dan *get* dengan menyertakan parameter nama kolom. Agar dapat menggunakan model ini maka pada fungsi *onCreate* harus mendefinisikan model ini dengan memanggil fungsi *registerSubclass*. Pendefinisian model ini dapat dilihat pada Gambar 4.7.

```
@Override
public void onCreate() {
    super.onCreate();

    ParseObject.registerSubclass(Hospital.class);
    ParseObject.registerSubclass(Poli.class);
    ParseObject.registerSubclass(Queue.class);

    Parse.enableLocalDatastore(this);
    Parse.initialize(this,
        "dohmliHLueuRlfBmV4gQho1sQ7ze3WvTrEUhgV85",
        "bb3vUtqkYVz2TsJ0OLIZhqbBZj5zWYFcXzen0Bv7");

    …
}
```

**Gambar 4.7 Kode Sumber Inisiasi Parse**

Kelas *Hospital* merupakan kelas model untuk tabel *Hospital* pada basis data. Kelas ini memiliki 3 fungsi yang diperlukan yaitu *getName* untuk mendapatkan nama dari rumah sakit, *getAddress* untuk mendapatkan alamat dari rumah sakit dan *getPictureUrl* untuk mendapatkan alamat url dari gambar rumah sakit. Kode sumber untuk kelas hospital terdapat pada Gambar 4.8.

```
@ParseClassName("Hospital")
public class Hospital extends ParseObject {
    public String getName() {
        return getString("name");
    }
    public String getAddress() {
        return getString("address");
    }
    public String getPictureUrl() {
        return getParseFile("picture").getUrl();
    }
}
```

**Gambar 4.8 Kode Sumber Kelas Hospital**

Kelas Poli merupakan kelas model untuk tabel Poli pada basis data. Kelas ini memiliki 5 fungsi yang diperlukan yaitu *getHospital* untuk mendapatkan obyek rumah sakit dimana poli ini berada, *getName* untuk mendapatkan nama dari poli, *getQueueNumber* untuk mendapatkan nomor antrian yang saat ini sedang dilayani, *getQueueTotal* untuk mendapatkan total nomor antrian yang juga berarti nomor antrian terakhir pada poli, *isOpen* untuk mendapatkan status apakah antrian pada poli buka ataukah tutup. Kode sumber untuk kelas poli terdapat pada Gambar 4.9.

Kelas *Queue* merupakan kelas model untuk tabel *Queue* pada basis data. Kelas ini memiliki 5 fungsi yang diperlukan yaitu *getPoli* untuk mendapatkan obyek poli, *getType* untuk mendapatkan tipe antrian dimana 0 berarti antrian diambil oleh pasien dan 1 berarti diambil oleh admin, *getNumber* untuk mendapatkan nomor antrian, *getCode* untuk mendapatkan kode antrian, *getStatus* untuk mendapatkan status apakah nomor antrian ini sudah dilayani dimana nilai 0 berarti belum dipanggil, nilai 1 berarti pasien hadir dan 2 berarti pasien tidak hadir. Kode sumber untuk kelas *queue* terdapat pada Gambar 4.10.

```
@ParseClassName("Poli")
public class Poli extends ParseObject {
    public Hospital getHospital() {
        return (Hospital) getParseObject("hospital");
    }
    public String getName() {
        return getString("name");
    }
    public int getQueueNumber() {
        return getInt("queueNumber");
    }
    public int getQueueTotal() {
        return getInt("queueTotal");
    }
    public boolean isOpen(){
        return getBoolean("open");
    }
}
```

**Gambar 4.9 Kode Sumber Kelas Poli**

```
@ParseClassName("Queue")
public class Queue extends ParseObject {
    public Poli getPoli(){
        return (Poli) getParseObject("poli");
    }
    public int getType() {
        return getInt("type");
    }
    public int getNumber() {
        return getInt("number");
    }
    public int getCode() {
        return getInt("code");
    }
    public int getStatus() {
        return getInt("status");
    }
}
```

**Gambar 4.10 Kode Sumber Kelas Queue**

### 4.2.3. Implementasi Aplikasi Android bagi Pasien

Activity utama pada aplikasi antrian ini adalah kelas MainActivity. Untuk pengaturan navigasi utama aplikasi ini menggunakan *Navigationdrawer*. Penulis menggunakan *NavigationDrawer* sebagai navigasi utama karena navigasi jenis ini sangat fleksibel dan dapat menampung banyak navigasi sehingga mempermudah pengembangan aplikasi ke depannya kelak. Pada bagian drawer ada 2 item navigasi yaitu navigasi untuk ke halaman daftar rumah sakit dan navigasi ke halaman antrian yang sudah diambil pasien. Pendefinisian menu drawer ini ada dalam res/menu/drawer.xml yang dapat dilihat pada Gambar 4.11.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:icon="@drawable/ic_hospital_24dp"
        android:id="@+id/hospital"
        android:checkable="true"
        android:title="@string/action_hospital"/>

    <item
        android:icon="@drawable/ic_queue_24dp"
        android:id="@+id/my_queue"
        android:checkable="true"
        android:title="@string/action_my_queue"/>

</menu>
```

**Gambar 4.11 Kode Sumber NavigationDrawer Menu**

#### 4.2.3.1. Implementasi Daftar Rumah Sakit bagi Pasien

Navigasi yang pertama akan mengarahkan pasien ke halaman daftar rumah sakit. Halaman ini didefinisikan pada kelas *HospitalListFragment*. Untuk mendapatkan semua daftar rumah sakit, aplikasi akan melakukan *query* ke Parse *server* dengan menggunakan obyek *ParseQuery*. Komputasi ini dilakukan pada

fungsi *getHospitals*. Fungsi *findInBackground* pada *ParseQuery* akan mengeksekusi *query* secara *asynchronous* sehingga tidak mengganggu tampilan. Selama query dieksekusi, antar muka akan menampilkan *ProgressBar* dengan memanggil fungsi *showProgress* yang menandakan aplikasi sedang mengambil data dari *server*. Hasil *query* berupa daftar rumah sakit yang akan ditampilkan menggunakan *RecyclerView* dengan memanggil fungsi *setData* pada *adapter*. Kode sumber fungsi ini dapat dilihat pada Gambar 4.12.

```
private void getHospitals(){
    ParseQuery<Hospital> query = ParseQuery.getQuery(Hospital.class);
    query.findInBackground(new FindCallback<Hospital>() {
        public void done(List<Hospital> list, ParseException e) {
            showProgress(false);

            if (e != null){
                Toast.makeText(getActivity(),
                    e.getMessage(), Toast.LENGTH_SHORT).show();

                return;
            }

            mAdapter.setData(list);
        }
    });

    showProgress(true);
}
```

**Gambar 4.12 Kode Sumber Fungsi *getHospitals* pada Kelas *HospitalListFrament***

Jika pasien menekan salah satu rumah sakit dari tampilan daftar rumah sakit maka aplikasi akan menampilkan daftar poli yang ada pada rumah sakit tersebut. Komputasi ini dilakukan dalam fungsi *showHospital*. Hal yang dilakukan pada fungsi ini adalah mendapatkan data rumah sakit yang bersangkutan dan mengirimkan *objectId* rumah sakit ini ke *fragment* baru yang akan

menampilkan daftar poli. Data rumah sakit disimpan di penyimpanan lokal dengan fungsi *pinInBackground*. Kode sumber fungsi dapat dilihat pada Gambar 4.13.

```
@OnClick(R.id.root)
public void onClick() {
    Hospital data = getData();
    data.pinInBackground();

    Bundle args = new Bundle();
    args.putString("hospital", data.getObjectId());

    Fragment fragment = new HospitalFragment();
    fragment.setArguments(args);

    MainActivity.replaceFragment(HospitalListFragment.this, fragment);
}
```

**Gambar 4.13 Kode Sumber Fungsi *showHospital* pada Kelas *HospitalListFragment***

### 4.2.3.2. Implementasi Daftar Poli bagi Pasien

Proses selanjutnya setelah pasien memilih rumah sakit adalah aplikasi mencari daftar poli yang ada pada rumah sakit tersebut. Komputasi ini dilakukan pada kelas *HospitalFragment*. Pada fungsi *onCreateView*, aplikasi akan mengambil data obyek *hospital* yang dikirimkan dari kelas *HospitalListFragment* dari penyimpanan local dengan memanggil fungsi *fetchFromLocalDatastoreInBackground*. Data rumah sakit yang didapat yaitu nama dan alamat rumah sakit akan ditampilkan pada bagian *toolbar* aplikasi. Aplikasi juga akan mengambil gambar rumah sakit dari Parse *server* melalui fungsi *displayImage* pada obyek *ImageLoader*. Kode sumber dapat dilihat pada Gambar 4.14.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    …

    mHospital = ParseObject.createWithoutData(
        Hospital.class, getArguments().getString("hospital"));

    mHospital.fetchFromLocalDatastoreInBackground(
        new GetCallback<Hospital>() {

        @Override
        public void done(Hospital object, ParseException e) {
            CollapsingToolbarLayout title = ((CollapsingToolbarLayout)
                view.findViewById(R.id.collapsing_toolbar));
            TextView address = (TextView)
                view.findViewById(R.id.header_detail);

            title.setTitle(mHospital.getName());
            address.setText(mHospital.getAddress());

            ImageLoader.getInstance()
                .displayImage(mHospital.getPictureUrl(), (ImageView)
                view.findViewById(R.id.header));
        }
    });

    …
}
```

**Gambar 4.14 Kode Sumber Fungsi *onCreateView* pada Kelas *HospitalFragment***

Untuk mendapatkan daftar poli pada rumah sakit, aplikasi akan melakukan *query* pada tabel Poli ke Parse *server*. Komputasi ini dilakukan pada fungsi *getPolis*. *Query* dibatasi pada rumah sakit yang dipilih dengan memanggil fungsi *whereEqualTo*. Setelah mendapatkan daftar poli, maka aplikasi akan melakukan *query* sekali lagi untuk mendapatkan daftar antrian yang sedang dimiliki pasien pada rumah sakit ini. Komputasi ini dilakukan pada fungsi *getQueues*. Kode sumber kedua fungsi ini dapat dilihat pada Gambar 4.15.

```
public void getPolis(){
    ParseQuery<Poli> query = ParseQuery.getQuery(Poli.class);
    query.whereEqualTo("hospital", mHospital);
    query.findInBackground(new FindCallback<Poli>() {
        public void done(List<Poli> list, ParseException e) {
            if (e == null) {
                mPolis = list;
                getQueues();
            } else {
                Toast.makeText(getActivity(), e.getMessage(),
                        Toast.LENGTH_SHORT).show();
            }
        }
    });
    showProgress(true);
}
private void getQueues(){
    if (ParseUser.getCurrentUser()== null){
        mAdapter.setData(mPolis);
        showProgress(false);
        return;
    }
    ParseQuery<Queue> query = ParseQuery.getQuery(Queue.class);
    query.whereEqualTo("createdBy", ParseUser.getCurrentUser());
    query.whereEqualTo("status", 0);
    query.whereEqualTo("type", 0);
    query.whereContainedIn("poli", mPolis);
    query.findInBackground(new FindCallback<Queue>() {
        public void done(List<Queue> list, ParseException e) {
            showProgress(false);
            if (e == null) {
                for (Queue queue : list) {
                    myQueues.put(queue.getPoli().getObjectId(), queue);
                }
                mAdapter.setData(mPolis);
            } else {
                Toast.makeText(getActivity(), e.getMessage(),
                        Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

**Gambar 4.15 Kode Sumber Fungsi *getPolis* dan *getQueue* pada Kelas *HospitalFragment***

Ketika pasien memilih poli dari daftar, aplikasi akan melihat terlebih dahulu apakah poli dalam status buka. Pasien tidak dapat memilih poli jika status poli adalah tutup kecuali dia memiliki nomor antrian di poli tersebut. Selanjutnya untuk menampilkan poli, aplikasi akan mengirimkan *objectId* dari poli tersebut. Komputasi ini dilakukan pada fungsi *showPoli* yang dapat dilihat pada Gambar 4.16.

```
@OnClick(R.id.root)
public void showPoli() {
    Poli poli = getData();

    if (!poli.isOpen() && queue != null){
        Toast.makeText(getActivity(), R.string.poli_closed,
                Toast.LENGTH_SHORT).show();
    } else {
        Intent intent = new Intent(getActivity(),
                PoliDispatchActivity.class);
        intent.putExtra("poli", poli.getObjectId());
        getActivity().startActivityForResult(intent, 0);
    }
}
```

**Gambar 4.16 Kode Sumber Fungsi *showPoli* pada Kelas *HospitalFragment***

### 4.2.3.3. Implementasi Antar Muka Poli bagi Pasien

Antar muka poli akan muncul setelah pasien memilih poli dari daftar poli rumah sakit. Untuk dapat masuk ke antar muka ini pasien diharuskan *login* ke dalam sistem terlebih dahulu. Di bagian ini aplikasi akan melakukan *query* ke Parse *server* untuk mendapatkan data mengenai poli yang bersangkutan beserta antrian yang dimiliki oleh pasien saat ini. Kedua *query* ini dilakukan pada fungsi *getPoli* dan *getQueue*. *getPoli* akan mencari data poli berdasarkan objectId dari poli yang dikirimkan oleh *HospitalFragment* sebelumnya. *getQueue* akan mencari

antrian pasien sesuai poli. Kedua fungsi ini dapat dilihat pada Gambar 4.17 .

```
private void getPoli(){
    ParseQuery<Poli> query = ParseQuery.getQuery(Poli.class);
    query.include("hospital");
    query.getInBackground(mPoliId, new GetCallback<Poli>() {
        @Override
        public void done(Poli poli, ParseException e) {
            mPoli = poli;
            getQueue();
        }
    });
    showProgress(true);
}

private void getQueue(){
    ParseQuery<Queue> query = ParseQuery.getQuery(Queue.class);
    query.whereEqualTo("createdBy", ParseUser.getCurrentUser());
    query.whereEqualTo("status", 0);
    query.whereEqualTo("type", 0);
    query.whereEqualTo("poli", mPoli);

    query.getFirstInBackground(new GetCallback<Queue>() {
        @Override
        public void done(Queue queue, ParseException e) {
            showProgress(false);
            if (e == null) {
                mQueue = queue;
            }
            updateUI();
        }
    });
}
```

**Gambar 4.17 Kode Sumber Fungsi *getPoli* dan *getQueue* pada Kelas *PoliActivity***

Pasien dapat melakukan pengambilan nomor antrian melalui halaman poli. Pengambilan nomor antrian ini dilakukan dengan memanggil fungsi *addQueue* yang ada pada Parse *Cloud Code*. Untuk melakukannya dapat menggunakan fungsi

*callFunctionInBackground* pada kelas *ParseCloud* dengan menyertakan parameter berupa *objectId* dari poli yang bersangkutan. Komputasi ini dilakukan dalam fungsi createQueue yang dapat dilihat pada Gambar 4.18.

```
@OnClick(R.id.queue_add)
protected void createQueue(){
    Map<String, Object> params = new HashMap<>();
    params.put("poli", mPoli.getObjectId());
    ParseCloud.callFunctionInBackground("addQueue", params,
         new FunctionCallback<Map<String,ParseObject>>() {
        @Override
        public void done(Map<String,ParseObject> result,
                 ParseException e) {
            showProgress(false);
            if (e == null) {
                mPoli = (Poli) result.get("poli");
                mQueue = (Queue) result.get("newQueue");
                updateUI();
            } else {
                Toast.makeText(PoliActivity.this, e.getMessage(),
                         Toast.LENGTH_SHORT).show();
            }
        }
    });
    showProgress(true);
}
```

**Gambar 4.18 Kode Sumber Fungsi *createQueue* pada Kelas *PoliActivity***

Pasien dapat menggunakan aplikasi ini tanpa perlu *login* atau mendaftar lebih dahulu. Pasien baru diharuskan untuk *login* saat mereka melakukan pengambilan nomor antrian. Parse menyediakan kelas *LoginDispatchActivity* untuk mengatasi hal ini. Kelas ini menjadi perantara antar 2 *activity* di mana *activity* pertama tidak mengharuskan pengguna untuk *login* sedangkan *activity* kedua mengharuskan pengguna untuk *login*. Dalam implementasi ini dibuat kelas *PoliDispatchActivity* yang merupakan *extends* dari kelas *LoginDispatchActivity*. Jika pengguna belum melakukan *login*, kelas ini akan memanggil

kelas yang digunakan untuk *login*. Jika pengguna sudah melakukan *login* maka kelas ini akan memanggil *PoliActivity*.

Setelah proses *login* selesai, kelas ini akan mendefinisikan obyek *ParseInstallation* dan menghubungkannya dengan obyek *ParseUser* dari hasil *login* tersebut. Obyek *ParseInstallation* ini memiliki data yang berguna untuk melakukan *push notification*. Kode sumber kelas *PoliDispatchActivity* dapat dilihat pada Gambar 4.19.

```
public class PoliDispatchActivity extends LoginDispatchActivity {
    @Override
    protected Class<?> getTargetClass() {
        return PoliActivity.class;
    }

    @Override
    protected void onActivityResult(int reqCode, int resCode,
         Intent data) {
        if (reqCode == LOGIN_REQUEST && resCode == RESULT_OK) {
            ParseInstallation pi =
                 ParseInstallation.getCurrentInstallation();
            pi.put("user", ParseUser.getCurrentUser());
            pi.saveInBackground();
        }
        super.onActivityResult(reqCode, resCode, data);
    }
}
```

**Gambar 4.19 Kode Sumber Kelas *PoliDispatchActivity***

### 4.2.3.4. Implementasi Notifikasi Giliran

Aplikasi Antrian untuk pasien akan menerima notifikasi jika nomor antrian yang telah diambil mendekati nomor antrian yang dipanggil di aplikasi admin. Parse telah menyediakan kelas *PushService* untuk melakukan *push notification.* Ada dua sumber yang bisa digunakan oleh *PushService* yaitu *Google Cloud Messaging* (GCM) atau jaringan push dari Parse sendiri. GCM hanya dapat digunakan jika dalam perangkat tersebut memiliki aplikasi *Google Play Store*. Karena tidak semua perangkat

android memilikinya maka pada implementasi ini digunakan kedua metode sumber. Jika dalam perangkat tersebut memiliki *Google Play Store* maka secara otomatis yang akan digunakan sebagai sumber adalah GCM, sedangkan jika tidak memiliki *Google Play Store* maka jaringan Parse yang digunakan. Untuk menggunakan *PushService* perlu mengkonfigurasi beberapa *permission* dalam AndroidManifest.xml. Beberapa permission ini antara lain *android.permission.internet* digunakan agar aplikasi dapat mengakses ke Parse server dan *android.permission.wake_lock* agar pesan tetap dapat diterima ketika *smartphone* tidak dalam kondisi *standby.* Konfigurasi yang diperlukan ditunjukkan dalam Gambar 4.20.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>

<permission android:protectionLevel="signature"
        android:name="${applicationId}.permission.C2D_MESSAGE"/>
<uses-permission android:name="${applicationId}.permission.C2D_MESSAGE"/>
<uses-permission
        android:name="com.google.android.c2dm.permission.RECEIVE"/>
```

**Gambar 4.20 Kode Sumber Konfigurasi *Permission* untuk *PushService***

Selanjutnya perlu mendeklarasikan *PushService, GcmBroadcastReceiver,* dan *ParsePushBroadcastReceiver* dalam *tag* application di *AndroidManifest*. *GcmBroadcastReceiver* bertujuan untuk mengatur penerimaan pesan terkirim dari *Google Cloud Messaging* ke aplikasi Android. *ParsePushBroadcastReceiver* digunakan untuk mengatur notifikasi yang akan muncul. Pada implementasi ini penulis membuat kelas *PushReceiver* yang merupakan *extends* dari *ParsePushBroadcastReceiver* sehingga deklarasi yang digunakan pada *AndroidManifest* adalah *PushReceiver*. Kode sumber deklarasi dapat dilihat pada Gambar 4.21.

```
<service android:name="com.parse.PushService" />

<receiver android:name=".PushReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="com.parse.push.intent.RECEIVE" />
        <action android:name="com.parse.push.intent.DELETE" />
        <action android:name="com.parse.push.intent.OPEN" />
    </intent-filter>
</receiver>

<receiver android:name="com.parse.GcmBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE"/>
        <action
          android:name="com.google.android.c2dm.intent.REGISTRATION"/>

        <category android:name="${applicationId}" />
    </intent-filter>
</receiver>

<meta-data android:name="com.parse.push.gcm_sender_id"
    android:value="id:685532636047" />
```

**Gambar 4.21 Kode Sumber Pendeklarasian *PushService***

Pada kelas *PushReceiver*, penulis menambahkan kode pada fungsi *onPushReceive* untuk mengatur tampilan notifikasi. *Push notification* yang dikirimkan oleh Parse *Cloud Code* berisikan informasi mentah dari poli dan antrian saat ini. Pada fungsi *onPushReceive* ini informasi tersebut akan ditata ulang sehingga lebih mudah dimengerti oleh pasien. Kode sumber kelas PushReceiver dapat dilihat pada Gambar 4.22.

```
public class PushReceiver extends ParsePushBroadcastReceiver {
    @Override
    protected void onPushReceive(Context context, Intent intent) {
        try {
            JSONObject data = new JSONObject(
                    intent.getStringExtra(KEY_PUSH_DATA));
            data.put("title", data.optString("hospitalName") + " - " +
                    data.optString("poliName"));

            int turn = data.optInt("turn");
            if (turn > 0) {
                data.put("alert",
                    context.getString(R.string.notif_call_text, turn));
            } else {
                data.put("alert",
                    context.getString(R.string.notif_call_text_now));
            }

            intent.putExtra(KEY_PUSH_DATA, data.toString());

            super.onPushReceive(context, intent);
        } catch (Exception e){
        }
    }
}
```

**Gambar 4.22 Kode Sumber Kelas *PushReceiver***

### 4.2.3.5. Implementasi Daftar Nomor Antrian Pasien

Saat pasien menekan notifikasi, pasien akan diarahkan ke halaman antrian yang telah diambil oleh pasien itu. Halaman ini juga dapat diakses dari *NavigationDrawer* pada *item* "My Queue". Komputasi halaman ini terdapat pada kelas *MyQueueFragment*. Pada kelas ini aplikasi akan melakukan *query* pada tabel *Queue* dengan mencari antrian yang memiliki status 0 yaitu antrian yang belum dipanggil oleh admin. *Query* ini dilakukan pada fungsi *getQueues*. Kode sumber fungsi *getQueues* dapat dilihat pada Gambar 4.23.

```
public void getQueues(){
    ParseQuery<Queue> query = ParseQuery.getQuery(Queue.class);
    query.whereEqualTo("createdBy", ParseUser.getCurrentUser());
    query.whereEqualTo("type", 0);
    query.whereEqualTo("status", 0);
    query.include("poli");
    query.findInBackground(new FindCallback<Queue>() {
        public void done(List<Queue> list, ParseException e) {
            showProgress(false);
            if (e != null) {
                Toast.makeText(getActivity(), e.getMessage(),
                        Toast.LENGTH_SHORT).show();
                return;
            }
            mAdapter.setData(list);
        }
    });
    showProgress(true);
}
```

**Gambar 4.23 Kode Sumber Fungsi *getQueues* pada Kelas *MyQueueFragment***

### 4.2.4. Implementasi Aplikasi Android bagi Admin

Aplikasi Antrian bagi admin rumah sakit mengharuskan admin untuk login terlebih dahulu. Hal ini berbeda dengan aplikasi bagi pasien. Untuk itu pada fungsi *onStart* di kelas *MainActivity*, aplikasi akan melakukan pengecekan status login dengan memanggil fungsi *getCurrentUser* pada kelas *ParseUser*. Fungsi ini akan mengembalikan nilai *null* jika pengguna belum *login* ke sistem. Kode sumber fungsi *onStart* terdapat pada Gambar 4.24.

```
protected void onStart() {
    super.onStart();
    ParseUser user = ParseUser.getCurrentUser();
    if (user == null){
        Intent intent = new ParseLoginBuilder(this).build();
        startActivityForResult(intent, REQUEST_LOGIN);
    } else {
        mAccountView.setVisibility(View.VISIBLE);
        mSignInButton.setVisibility(View.GONE);

        mAccountName.setText(user.getString("name"));
        mAccountEmail.setText(user.getEmail());
    }
}
```

**Gambar 4.24 Kode Sumber Fungsi *onStart* pada Kelas *MainActivity***

Pada awal aplikasi dibuka, aplikasi akan mencari daftar rumah sakit yang memiliki relasi dengan admin. Daftar ini dapat diperoleh dengan melakukan *query* pada kelas *AdminHospital* pada Parse Basis Data. Proses ini dilakukan dalam fungsi *getHospitals* pada kelas *HospitalListFragment*. Jika hasil *query* hanya terdiri dari 1 rumah sakit, maka fungsi ini akan menampilkan halaman daftar poli pada rumah sakit itu. Jika hasil query lebih dari 1 rumah sakit maka fungsi ini akan menampilkan daftar rumah sakit yang bersangkutan. Kode sumber fungsi *getHospitals* dapat dilihat pada Gambar 4.25.

#### 4.2.4.1. Implementasi Daftar Rumah Sakit dan Daftar Poli bagi Admin

Tampilan daftar rumah sakit dan daftar poli pada admin hampir sama dengan pada pasien. Pada daftar poli yang membedakan antara admin dan pasien adalah pada bagian admin tidak perlu mencari nomor antrian karena pada dasarnya admin tidak memiliki nomor antrian. Pada bagian ini aplikasi hanya melakukan query untuk mendapatkan daftar poli saja.

```
private void getHospitals(){
    ParseQuery<ParseObject> queryAdmin =
       ParseQuery.getQuery("AdminHospital");
    queryAdmin.whereEqualTo("admin", ParseUser.getCurrentUser());
    queryAdmin.include("hospital");
    queryAdmin.findInBackground(new FindCallback<ParseObject>() {
        @Override
        public void done(List<ParseObject> list, ParseException e) {
            showProgress(false);
            if (e != null || list.size() < 1){
                Toast.makeText(getActivity(), "Error",
                    Toast.LENGTH_SHORT).show();
            } else if (list.size() == 1) {
                Hospital hospital = (Hospital) list.get(0)
                        .getParseObject("hospital");
                showHospital(hospital, false);
            } else {
                List<Hospital> hospitals = new ArrayList<>();
                for (ParseObject po : list) {
                    hospitals.add((Hospital)
                           po.getParseObject("hospital"));
                }
                mAdapter.setData(hospitals);
            }
        }
    });

    showProgress(true);
}
```

**Gambar 4.25 Kode Sumber Fungsi *getHospital* pada Kelas *HospitalListFragment***

### 4.2.4.2. Implementasi Antar Muka Poli bagi Admin

Tampilan antar muka poli bagi admin lebih kompleks dari pada pasien. Aplikasi akan melakukan *query* ke Parse *server* untuk mendapatkan data mengenai poli yang bersangkutan dan antrian yang dipanggil saat ini. Kedua *query* ini dilakukan pada fungsi *getPoli* dan *getQueue*. *getPoli* akan mencari data poli berdasarkan objectId dari poli yang dikirimkan oleh

*HospitalFragment*. *getQueue* akan mencari antrian yang dipanggil saat ini. Setelah mendapatkan data poli dan antrian, aplikasi akan menampilkan informasi pada antar muka sesuai kondisi data. Kedua fungsi ini dapat dilihat pada Gambar 4.26.

```
private void getPoli(){
    mPoli = ParseObject.createWithoutData(Poli.class, mPoliId);
    mPoli.fetchInBackground(new GetCallback<Poli>() {
        @Override
        public void done(Poli poli, ParseException e) {
            getQueue();
        }
    });
    showProgress(true);
}

private void getQueue(){
    ParseQuery<Queue> query = ParseQuery.getQuery(Queue.class);
    query.whereEqualTo("poli", mPoli);
    query.whereEqualTo("number", mPoli.getQueueNumber());
    query.getFirstInBackground(new GetCallback<Queue>() {
        @Override
        public void done(Queue queue, ParseException e) {
            showProgress(false);
            mCurrentQueue = queue;
            updateUI();
        }
    });
}
```

**Gambar 4.26 Kode Sumber *getPoli* dan *getQueue* pada Kelas *PoliActivity***

Admin dapat membuka antrian, menutup antrian, memanggil nomor antrian, dan mengambil nomor antrian. Keempat fitur ini terdapat pada fungsi *command*. Fungsi ini akan memanggil fungsi pada Parse Cloud Code. Kode sumber fungsi command dapat dilihat pada Gambar 4.27.

```
private void command(String method){
    showProgress(true);
    Map<String, Object> params = new HashMap<>();
    params.put("poli", mPoli.getObjectId());
    params.put("type", 1);

    ParseCloud.callFunctionInBackground(method, params,
       new FunctionCallback<Map<String,ParseObject>>() {
        @Override
        public void done(Map<String,ParseObject> result,
              ParseException e) {

            showProgress(false);
            if (e != null) {
                Toast.makeText(PoliActivity.this, e.getMessage(),
                    Toast.LENGTH_SHORT).show();
                return;
            }
            Poli poli = (Poli) result.get("poli");
            if (poli != null) {
                mPoli = poli;
            }
            Queue queue;
            if ((queue = (Queue) result.get("queue")) != null) {
                mCurrentQueue = queue;
            }
            if ((queue = (Queue) result.get("newQueue")) != null){
                mNewQueue = queue;
            }
            updateUI();
        }
    });
}
```

**Gambar 4.27 Kode Sumber Fungsi *command* pada Kelas *PoliActivity***

Admin diharuskan mengubah status kehadiran pasien setelah melakukan pemanggilan nomor antrian. Proses ini dilakukan pada fungsi *status*. Fungsi ini akan memanggil fungsi untuk mengubah status kehadiran di Parse Cloud Code. Kode sumber fungsi dapat dilihat pada Gambar 4.28.

```
private void status(int i){
    showProgress(true);
    Map<String, Object> params = new HashMap<>();
    params.put("queue", mCurrentQueue.getObjectId());
    params.put("status", i);

    ParseCloud.callFunctionInBackground("statusQueue", params,
       new FunctionCallback<Queue>() {
        @Override
        public void done(Queue result, ParseException e) {
            showProgress(false);
            if (e != null) {
                Toast.makeText(PoliActivity.this, e.getMessage(),
                    Toast.LENGTH_SHORT).show();
                return;
            }
            mCurrentQueue = result;
            updateUI();
        }
    });
}
```

**Gambar 4.28 Kode Sumber Pemanggilan Fungsi Status pada Parse Cloud Code**

# BAB V
# PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas mengenai hasil pengujian dan analisa kinerja yang telah dilakukan. Uji coba akan dibagi menjadi tiga bagian yaitu uji coba fungsionalitas, uji coba skenario dan uji coba performa.

## 5.1. Lingkungan Pengujian

Dalam melakukan pengujian terhadap aplikasi antrian ini digunakan satu buah smartphone android Infinix hot 2 dengan spesifikasi sebagai berikut.

- Prosesor Quad Core 1.3 Ghz
- RAM 2 GB
- Ukuran layar 5 inchi
- Memori penyimpanan 16 GB
- Sistem operasi Android Marsmallow Versi 6.0

## 5.2. Pengujian Fungsionalitas

Pada pengujian fungsionalitas akan diuji fungsi yang telah dibuat dalam sistem. Fungsi yang diuji dalam sistem dimulai dari fungsi pengambilan nomor oleh pasien, membuka antrian pada poli, pemanggilan nomor antrian disertai notifikasi di pasien, pengambilan nomor oleh admin, perubahan status kehadiran pasien, dan menutup antrian pada poli.

### 5.2.1. Pengujian Fungsionalitas Pengambilan Nomor Antrian oleh Pasien

Fungsionalitas utama pada aplikasi antrian untuk pasien adalah pengambilan nomor antrian. Uji coba fungsionalitas ini akan mengikuti prosedur yang ada pada Tabel 5.1 berikut.

**Tabel 5.1 Prosedur Uji Coba Fungsionalitas Pengambilan Nomor Antrian oleh Pasien**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengambilan Nomor oleh Pasien |
| Kode | UJ-01 |
| Tujuan Pengujian | Menguji fitur pengambilan nomor antrian pada poli oleh pasien |
| Kondisi Awal | Pasien sudah login ke aplikasi dan antrian pada poli sudah dalam keadaan *open*. |
| Data Input | - |
| Prosedur Pengujian | 1. Pasien memilih rumah sakit dan poli yang dikehendaki<br>2. Pasien menekan tombol *Get* untuk mengambil nomor antrian |
| Hasil yang Diharapkan | Pasien mendapatkan nomor antrian pada poli yang bersangkutan |
| Hasil yang Diperoleh | Pasien mendapatkan nomor antrian pada poli yang bersangkutan |

Pengujian ini ditujukan untuk menguji fungsionalitas pengambilan nomor antrian oleh pasien. Pasien pertama-tama memilih rumah sakit dan poli. Dalam uji coba ini yang dipilih adalah Poli Umum pada RSUD Dr Soetomo. Sistem akan mengambil data antrian pada poli dan menampilkan pada halaman poli. Bila pasien belum mengambil nomor antrian maka akan ada tombol *Get* untuk mengambil nomor antrian. Tampilan poli sebelum pengambilan nomor antrian dapat dilihat pada Gambar 5.1.


**Gambar 5.1 Tampilan Poli Sebelum Pengambilan Nomor Antrian (Kondisi Awal)**

Dengan menekan tombol *Get* maka aplikasi akan memanggil fungsi addQueue di Parse Cloud Code dan mengembalikan data antrian yang telah diambil. Hasil data nomor antrian berupa nomor antrian, tanggal pengambilan, dan kode antrian selanjutnya ditampilkan pada halaman poli. Sebagai tambahan untuk mempermudah pasien maka ditampilkan pula angka penunjuk berapa antrian lagi gilirannya tiba. Tampilan poli setelah pengambilan nomor dapat dilihat pada Gambar 5.2.


**Gambar 5.2 Tampilan Poli Setelah Pengambilan Nomor Antrian (Hasil Uji)**

### 5.2.2. Pengujian Fungsionalitas Pembukaan Antrian pada Poli

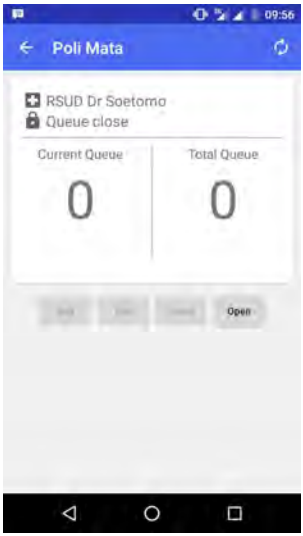
Antrian pada poli memiliki status *open* dan *close*. Pasien hanya dapat mengambil nomor antrian pada poli yang memiliki status *open*. Aplikasi pada admin dapat mengubah status antrian ini. Pada bagian ini yang akan diuji adalah fungsionalitas pembuka antrian pada poli. Prosedur pengujiannya dijelaskan pada Tabel 5.2.

**Tabel 5.2 Prosedur Uji Coba Fungsionalitas Pembukaan Antrian pada Poli**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pembukaan Antrian pada Poli |
| Kode | UJ-02 |
| Tujuan Pengujian | Menguji fitur untuk mengubah status antrian pada poli menjadi *open* |
| Kondisi Awal | Admin sudah login ke aplikasi, memiliki hak akses terhadap rumah sakit yang bersangkuta, dan status antrian pada poli adalah *close*. |
| Data Input | - |
| Prosedur Pengujian | 1. Admin memilih poli dengan status *close*.<br>2. Admin menekan tombol *Open* untuk membuka antrian pada poli. |
| Hasil yang Diharapkan | Status antrian pada poli menjadi *open* |
| Hasil yang Diperoleh | Status antrian pada poli menjadi *open* |

Proses pengujian fungsionalitas pembuka antrian pada poli dilakukan di aplikasi admin. Admin harus sudah dalam keadaan login ke dalam aplikasi dan memiliki hak akses terhadap rumah sakit yang bersangkutan. Selanjutnya admin memilih poli dengan status *close*. Pada halaman poli, hanya tombol *Open* yang dapat ditekan karena status antrian adalah *close*. Tampilan poli dapat dilihat pada Gambar 5.3.

Selanjutnya admin menekan tombol *Open* untuk membuka antrian pada poli. Aplikasi akan memanggil fungsi *openQueue* pada Parse *Cloud Code* dan mengembalikan hasil berupa data poli dengan status *open*. Tampilan halaman poli akan berubah karena status antrian sekarang menjadi *open* yang ditunjukkan oleh Gambar 5.4.




**Gambar 5.3 Tampilan Poli dengan Status Antrian *Close* (Kondisi Awal)**


**Gambar 5.4 Tampilan Poli dengan Status Antrian *Open* (Hasil Uji)**

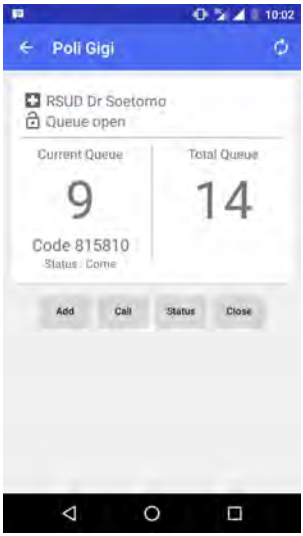
### 5.2.3. Pengujian Fungsionalitas Penutupan Antrian pada Poli

Antrian pada poli memiliki status *close* menandakan pasien tidak dapat mengambil nomor antrian pada poli tersebut. Pada bagian ini yang akan diuji adalah fungsionalitas penutupan antrian pada poli. Prosedur pengujiannya dijelaskan pada Tabel 5.3.

**Tabel 5.3 Prosedur Uji Coba Fungsionalitas Penutupan Antrian pada Poli**

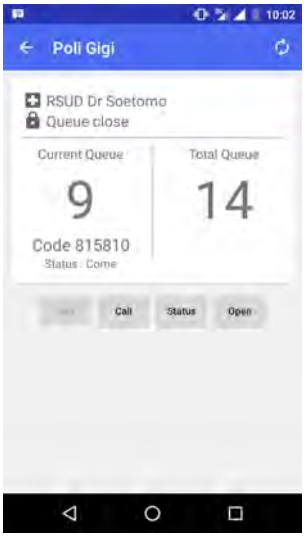
| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Penutup Antrian pada Poli |
| Kode | UJ-03 |
| Tujuan Pengujian | Menguji fitur untuk mengubah status antrian pada poli menjadi *close* |
| Kondisi Awal | Admin sudah login ke aplikasi, memiliki hak akses terhadap rumah sakit yang bersangkuta, dan status antrian pada poli adalah *open*. |
| Data Input | - |
| Prosedur Pengujian | 1. Admin memilih poli dengan status *open*.<br>2. Admin menekan tombol *Close* untuk menutup antrian pada poli. |
| Hasil yang Diharapkan | Status antrian pada poli menjadi *close* |
| Hasil yang Diperoleh | Status antrian pada poli menjadi *close* |

Proses pengujian fungsionalitas penutupan antrian pada poli dilakukan di aplikasi admin. Admin harus sudah dalam keadaan login ke dalam aplikasi dan memiliki hak akses terhadap rumah sakit yang bersangkutan. Selanjutnya admin memilih poli dengan status *open*. Tampilan poli dapat dilihat pada Gambar 5.5.




**Gambar 5.5 Tampilan Poli dengan Status Antrian *Open* (Kondisi Awal)**

Selanjutnya admin menekan tombol *Close* untuk menutup antrian pada poli. Aplikasi akan memanggil fungsi *closeQueue* pada Parse *Cloud Code* dan mengembalikan hasil berupa data poli dengan status *close*. Pada antrian dengan status *close*, admin masih bisa melakukan panggilan nomor antrian selama masih ada nomor antrian pada poli tersebut. Tampilan halaman poli akan berubah karena status antrian sekarang menjadi *close* yang ditunjukkan oleh Gambar 5.6.




**Gambar 5.6 Tampilan Poli dengan Status Antrian *Close* setelah Pengujian Penutupan Antrian (Hasil Uji)**

### 5.2.4. Pengujian Fungsionalitas Pemanggilan Nomor Antrian

Admin dapat memanggil nomor antrian pada poli selama dalam poli tersebut masih ada nomor antrian yang belum dipanggil. Pada bagian ini yang akan diuji adalah fungsionalitas pemanggilan nomor antrian pada poli. Prosedur pengujiannya dijelaskan pada Tabel 5.4.

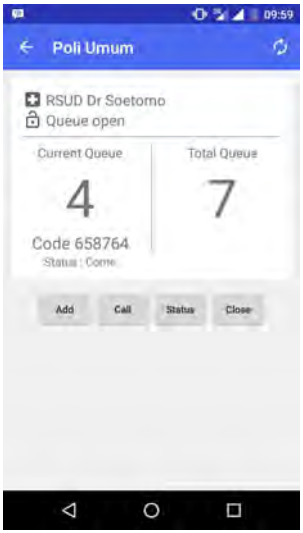
**Tabel 5.4 Prosedur Uji Coba Fungsionalitas Penutupan Antrian**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pemanggilan Nomor Antrian |
| Kode | UJ-04 |
| Tujuan Pengujian | Menguji fitur untuk memanggil nomor antrian berikutnya. |
| Kondisi Awal | Admin sudah login ke aplikasi, memiliki hak |

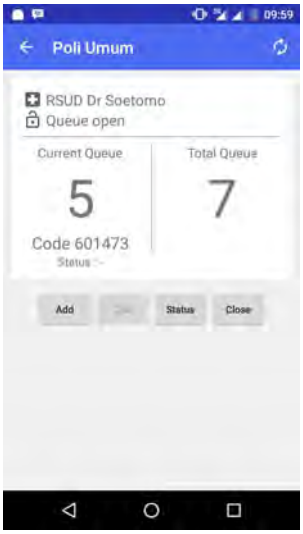
| | |
|---|---|
| | akses terhadap rumah sakit yang bersangkutan, dan antrian pada poli memiliki nomor antrian yang belum terpanggil. |
| Data Input | - |
| Prosedur Pengujian | 1. Admin memilih poli yang memiliki nomor antrian yang belum terpanggil.<br>2. Admin menekan tombol *Call* untuk memanggil nomor antrian berikutnya. |
| Hasil yang Diharapkan | Nomor antrian yang dipanggil pada poli berubah ke nomor antrian berikutnya dan notifikasi pada aplikasi pasien yang telah mengambil nomor antrian pada poli yang bersangkutan |
| Hasil yang Diperoleh | Nomor antrian yang dipanggil pada poli berubah ke nomor antrian berikutnya dan notifikasi pada aplikasi pasien yang telah mengambil nomor antrian pada poli yang bersangkutan |

Proses pengujian fungsionalitas pemanggilan nomor antrian pada poli dilakukan di aplikasi admin. Admin harus sudah dalam keadaan login ke dalam aplikasi dan memiliki hak akses terhadap rumah sakit yang bersangkutan. Selanjutnya admin memilih poli yang memiliki nomor antrian yang belum terpanggil, yaitu dengan ditandai dengan nilai *totalQueue* lebih besar dari *currentQueue*. Tampilan poli sebelum proses pengujian dapat dilihat pada Gambar 5.7.

Selanjutnya admin menekan tombol *Call* untuk memanggil nomor antrian berikutnya, yaitu nomor antrian 5. Aplikasi akan memanggil fungsi *callQueue* pada Parse *Cloud Code* dan mengembalikan hasil berupa data poli dan data queue untuk nomor yang terpanggil. Data queue berupa nomor antrian dan kode antrian akan ditampilkan pada layar, ditunjukkan oleh Gambar 5.8.

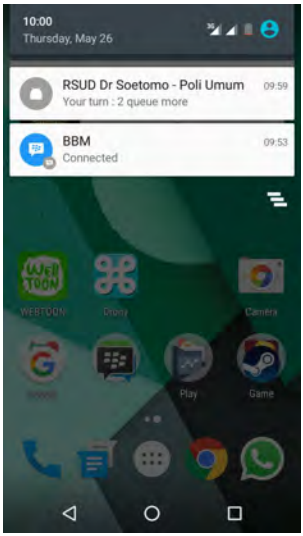



**Gambar 5.7 Tampilan Poli yang Memiliki Nomor Antrian yang Belum Terpanggil (Kondisi Awal)**




**Gambar 5.8 Tampilan Poli Setelah Pemanggilan Nomor Antrian (Hasil Uji)**

Proses pemanggilan nomor antrian ini merupakan trigger pada Parse Server untuk mengirimkan notifikasi ke nomor antrian yang terpanggil. Fungsi *callQueue* juga melakukan proses pengiriman push notification ke nomor antrian yang terpanggil dan 4 nomor selanjutnya. Pada perangkat smartphone pasien yang sebelumnya mengambil nomor antrian akan mendapatkan notifikasi seperti yang terlihat pada Gambar 5.9.




**Gambar 5.9 Tampilan Notifikasi Giliran Antrian pada Pasien (Hasil Uji)**

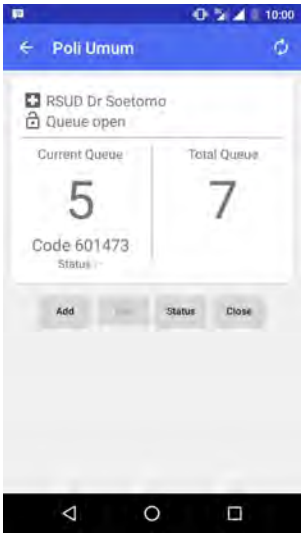
### 5.2.5. Pengujian Fungsionalitas Pengubahan Status Kehadiran Pasien

Setelah pemanggilan nomor antrian, tugas admin selanjutnya adalah mengubah status kehadiran pasien yang bersangkutan. Pada bagian ini yang akan dilakukan pengujian terhadap fungsionalitas pengubahan status kehadiran pasien. Prosedur pengujiannya dijelaskan pada Tabel 5.5.

**Tabel 5.5 Prosedur Uji Coba Fungsionalitas Pengubahan Status Kehadiran Pasien**

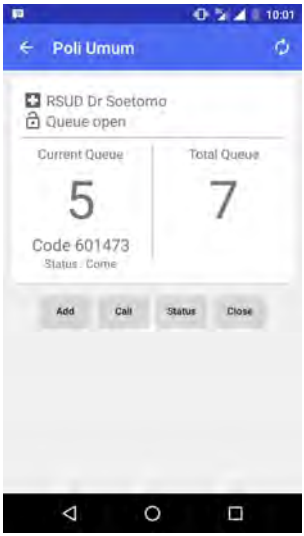
| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengubahan Status Kehadiran Pasien |
| Kode | UJ-05 |
| Tujuan Pengujian | Menguji fitur untuk mengubah status kehadiran pasien terpanggil. |
| Kondisi Awal | Admin sudah login ke aplikasi, memiliki hak akses terhadap rumah sakit yang bersangkutan. |
| Data Input | - |
| Prosedur Pengujian | 1. Admin melakukan pemanggilan pada antrian poli.<br>2. Admin menekan tombol *Status* untuk mengubah status kehadiran nomor antrian yang terpanggil. |
| Hasil yang Diharapkan | Nomor antrian yang terpanggil berubah statusnya sesuai kehadiran yang dipilih |
| Hasil yang Diperoleh | Nomor antrian yang terpanggil berubah statusnya sesuai kehadiran yang dipilih |

Proses pengujian fungsionalitas pengubahan status kehadiran pasien dilakukan di aplikasi admin. Admin harus sudah dalam keadaan login ke dalam aplikasi dan memiliki hak akses terhadap rumah sakit yang bersangkutan. Selanjutnya admin memilih poli dan melakukan pemanggilan nomor antrian. Nomor antrian yang terpanggil belum memiliki status kehadiran, seperti yang terlihat pada Gambar 5.10.




**Gambar 5.10 Tampilan Poli Setelah Pemanggilan Nomor Antrian (Kondisi Awal)**

Selanjutnya dengan menekan tombol *Status*, admin diberikan 2 pilihan apakah pasien yang bersangkutan hadir atau tidak. Aplikasi akan memanggil fungsi *statusQueue* pada Parse *Cloud Code* dan mengembalikan hasil berupa data antrian sesuai status yang dipilih. Tampilan status nomor antrian yang terpanggil akan berubah sesuai kehadiran yang dipilih, seperti yang terlihat dalam Gambar 5.11.




**Gambar 5.11 Tampilan Poli Setelah Pengubahan Status Kehadiran (Hasil Uji)**

### 5.2.6. Pengujian Fungsionalitas Pengambilan Nomor Antrian oleh Admin

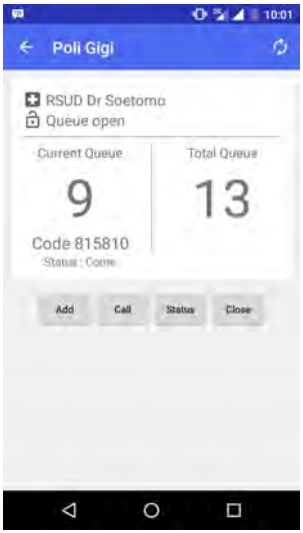
Admin juga dapat mengambil nomor antrian pada poli. Fungsionalitas ini ditujukan untuk pasien yang mengambil antrian langsung di rumah sakit yang bersangkutan. Pada bagian ini yang akan dilakukan pengujian terhadap fungsionalitas pengambilan nomor antrian oleh admin. Prosedur pengujiannya dijelaskan pada Tabel 5.6.

**Tabel 5.6 Prosedur Uji Coba Fungsionalitas Pengambilan Nomor Antrian oleh Admin**

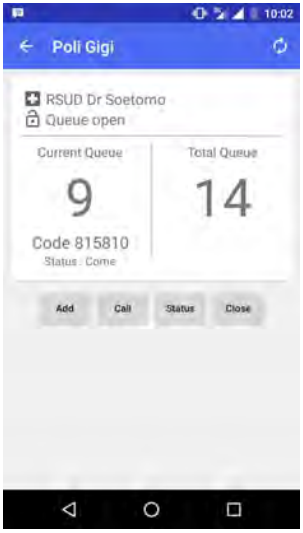
| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengambilan Nomor Antrian oleh Admin |
| Kode | UJ-05 |
| Tujuan Pengujian | Menguji fitur untuk mengambil nomor antrian dengan menggunakan aplikasi admin |
| Kondisi Awal | Admin sudah *login* ke aplikasi, memiliki hak akses terhadap rumah sakit yang bersangkutan, dan poli dalam status *open* |
| Data Input | - |
| Prosedur Pengujian | 1. Admin memilih poli.<br>2. Admin menekan tombol *Add* untuk mengambil nomor antrian pada poli tersebut. |
| Hasil yang Diharapkan | *TotalQueue* akan berubah sesuai nomor antrian yang diambil |
| Hasil yang Diperoleh | *TotalQueue* akan berubah sesuai nomor antrian yang diambil |

Pengujian ini ditujukan untuk menguji fungsionalitas pengambilan nomor antrian oleh admin. Admin harus sudah dalam keadaan login ke dalam aplikasi dan memiliki hak akses terhadap rumah sakit yang bersangkutan. Selanjutnya admin memilih poli dengan status *open*. Tampilan poli sebelum pengambilan nomor antrian dapat dilihat pada Gambar 5.12.

Dengan menekan tombol *Add* maka aplikasi akan memanggil fungsi *addQueue* di Parse *Cloud Code* dan mengembalikan data antrian yang telah diambil. Tampilan poli setelah pengambilan nomor dapat dilihat pada Gambar 5.13.




**Gambar 5.12 Tampilan Poli Sebelum Pengambilan Nomor Antrian oleh Admin (Kondisi Awal)**




**Gambar 5.13 Tampilan Poli Setelah Pemanggilan Nomor Antrian oleh Admin (Hasil Uji)**

*[Halaman ini sengaja dikosongkan]*

# BAB VI
# KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

## 6.1. Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

- Sistem ini dapat memudahkan pasien dalam mengambil nomor antrian karena pasien tidak perlu datang ke rumah sakit untuk mengambil nomor antrian berdasarkan hasil uji coba fungsionalitas pengambilan nomor antrian oleh pasien.
- Sistem ini berhasil melakukan pengambilan nomor antrian dan pemanggilan nomor antrian dengan akurat berdasarkan hasil uji coba fungsionalitas pemanggilan nomor antrian.

## 6.2. Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

- Memindahkan Parse server ke server sendiri sehingga respon server terhadap permintaan aplikasi menjadi lebih cepat
- Fungsionalitas sistem dapat dikembangkan dengan menambahkan pendeteksian lokasi pasien sehingga dapat mencari rumah sakit terdekat.
- Penambahan aplikasi untuk pengambilan nomor antrian bagi pasien yang datang langsung ke rumah sakit.

- Pengambilan nomor antrian oleh admin disinkronisasi dengan pengumuman nomor yang terpanggil.

# DAFTAR PUSTAKA


[1] "Queue area - Wikipedia, the free encyclopedia," [Online]. Available: https://en.wikipedia.org/wiki/Queue_area. [Accessed 13 September 2015].

[2] I. Sommerville, Software Engineering, Boston: Addison-Wesley, 2007.

[3] K. Lane, "Backend as a Service (BaaS)," [Online]. Available: http://baas.apievangelist.com/.

[4] "Android Developers Guide | Parse," [Online]. Available: https://parse.com/docs/android/guide. [Accessed 2 Juni 2016].

[5] Y. Nugroho, March 2013. [Online]. Available: http://kur2003.if.itb.ac.id/file/IF2281_Java_API.pdf.

[6] M. Rouse, March 2014. [Online]. Available: http://searchmobilecomputing.techtarget.com/definition/push-notification.

[7] Android, Januari 2014. [Online]. Available: https://developer.android.com/google/gcm/gcm.html.

*[Halaman ini sengaja dikosongkan]*

# BIODATA PENULIS

Penulis, Kukuh Hannugroho lahir di Lumajang, Jawa Timur, pada tanggal 9 Januari 1990. Penulis adalah anak ke 5 dari 5 bersaudara.

Penulis menempuh pendidikan formal di SD Negeri 1 Pasirian (1996-2002), SMP Negeri 1 Pasirian (2002-2005) dan SMA Negeri 2 Lumajang (2005-2008). Pada tahun 2008, penulis menempuh pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember, Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Berbasis Jaringan dan aktif mengikuti kegiatan seminar yang diadakan oleh Jurusan dan Himpunan Mahasiswa Teknik Computer-Informatika (HMTC). Penulis dapat dihubungi melalui alamat email kukuhhannugroho@gmail.com