**PENGEMBANGAN *FRONT-END* DAN *BACK-END APPLICATION* UNTUK SISTEM KAMPUS PINTAR UBEACON**

**TUGAS AKHIR**

**oleh**

**RIZKY INDRA SYAFRIAN**

**NIM 13212049**

****

**PROGRAM STUDI TEKNIK ELEKTRO**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2016PENGEMBANGAN *FRONT-END* DAN *BACK-END APPLICATION* UNTUK SISTEM KAMPUS PINTAR UBEACON**

oleh

**Rizky Indra Syafrian**

Tugas Akhir ini telah diterima dan disahkan  
sebagai persyaratan untuk memperoleh gelar  
**SARJANA TEKNIK**  
di  
PROGRAM STUDI TEKNIK ELEKTRO  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG

Bandung, 16 Juni 2016  
Disetujui oleh

Pembimbing I,

**Ir. Emir Mauludi Husni, M.Sc., Ph.D.  
NIP 196707072006041016**

Pembimbing II,

**Andi Sama  
CIO PT Sinergi Wahana Gemilang**

**ABSTRAK**

**PENGEMBANGAN *FRONT-END* DAN *BACK-END APPLICATION* UNTUK SISTEM KAMPUS PINTAR UBEACON**

oleh

**RIZKY INDRA SYAFRIAN**

**NIM 13212049**

**PROGRAM STUDI TEKNIK ELEKTRO**

uBeacon (*University Beacon*) merupakan sistem kampus pintar yang dirancang untuk mempermudah kehidupan mahasiswa di kampus. Pada sistem uBeacon, *cloud* PaaS dan aplikasi yang berjalan di dalamnya berperan sebagai server yang melayani aplikasi *smartphone* uBeacon. Ketika aplikasi *smartphone* berinteraksi dengan sinyal BLE *beacon*, sebuah aksi yang disebut “*events*” terjadi. Pada sistem ini, *events* yang terjadi melibatkan pemrosesan data dan pertukaran data agar informasi dapat disampaikan ke aplikasi *smartphone*. Pemrosesan data ditangani oleh *back-end application* yang berjalan pada server. Pertukaran data dilayani oleh *database* pada server. Fungsi-fungsi dari sistem uBeacon juga membutuhkan seorang administrator yang bertugas untuk mengelola *database* informasi *beacon*, mengirimkan notifikasi, memonitor absensi kelas, dan memonitor lalu lintas pengguna aplikasi dengan laporan dan analitik yang dibuat oleh IBM Presence Insights. Tugas administrator akan dibantu oleh sebuah *front-end application*, meskipun beberapa tugas tetap mengharuskan administrator untuk bekerja langsung pada *back-end* dan *database*. Buku tugas akhir ini membahas tentang proses pembuatan dan langkah yang diambil pada saat pengembangan *front-end application* dan *back-end application* dari sistem uBeacon.

**Kata kunci:** *beacon*; *platform as a service*; *back-end*; *front-end*; *database*; IBM Bluemix; IBM Presence Insights.

**ABSTRACT**

**FRONT-END AND BACK-END APPLICATION DEVELOPMENT FOR UBEACON SMART CAMPUS SYSTEM**

by

**RIZKY INDRA SYAFRIAN**

**NIM 13212049**

**ELECTRICAL ENGINEERING MAJOR**

uBeacon (University Beacon) is a smart campus system designed to facilitate student life on campus. In uBeacon systems, cloud PaaS and applications running in it acts as a smartphone application server that serves uBeacon. When the smartphone application interacts with BLE beacon signal, an action called events occur. In this system, the events that occurred involving the processing of data and data exchange so that information can be delivered to the smartphone application. Data processing handled by back-end application running on the server. Data exchange is served by the database on the server. The functions of the uBeacon system also requires an administrator whose job is to manage the database information beacon, sending notifications, monitor class attendance, and monitor user traffic and analytic applications with reports made by IBM Presence Insights. The task of the administrator will be assisted by a front-end application, although some tasks still require an administrator to work directly on the back-end and database. This final project book focuses on the production process and the steps taken during the development of front-end applications and back-end application of the system uBeacon.

**Keywords:** beacon; platform as a service; back-end; front-end; database; IBM Bluemix; IBM Presence Insights.

**PRAKATA**

Puji syukur penulis panjatkan kehadirat Allah SWT yang atas rahmat dan karunia-Nya penulis dapat menyelesaikan tugas akhir yang berjudul “Pengembangan *Front-end* dan *Back-end Application* untuk Sistem Kampus Pintar uBeacon”.

Buku tugas akhir ini diselesaikan sebagai upaya pemenuhan syarat kelulusan dan memperoleh gelar Sarjana Teknik dari Program Studi Teknik Elektro di Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Selama melaksanakan tugas akhir ini, penulis mendapatkan bantuan, dukungan, dan dampingan dari berbagai pihak. Untuk itu, penulis ingin mengucapkan terima kasih kepada:

* Ayah, Ibu, dan semua keluarga besar yang telah memberikan dukungan moral kepada penulis selama menuntut ilmu di Institut Teknologi Bandung.
* Bapak Ir. Emir Mauludi Husni, M.Sc., Ph.D. selaku dosen pembimbing penulis atas bimbingan dan nasihatnya selama pengerjaan tugas akhir.
* Bapak Arief Sasongko, S.T., M.Sc., Ph.D. selaku ketua program studi Teknik Elektro ITB beserta Tim Tugas Akhir Teknik Elektro ITB yang telah memberikan arahan selama pengerjaan tugas akhir.
* Astari Purnomo dan Adirga Ibrahim Khairy selaku rekan satu tim tugas akhir yang telah bersedia bekerja sama dan berbagi suka duka dengan penulis selama pengerjaan tugas akhir.
* Teman-teman Ruang Riset Mandiri Basement Labtek VII (Goa Depan); Sam, Iren, Vivi (VISCar); Andas, Arfie, Nadia (Nyilem 3.0); Meynard, Adil, Afdhal (Cubesat); Riksa, Ipul, Hilmy (Illegal Fishing); yang telah menjadi teman satu residensi yang memberikan banyak pelajaran hidup.
* Teman-teman Ruang Laboratorium Kontrol Dasar dan Komputer Labtek VII (Goa Belakang) yang ribut dan tidak perlu disebutkan.
* Teman-teman Nomaden; Rendy, Fadhil, Adit (UAV CRN); Zuhdi; Alfi; Aris (ACWS); dan Ricky (e-Health); yang sering main ke residensi penulis dan membuat Tugas Akhir I menjadi menyenangkan.
* Teman-teman Teknik Elektro ITB 2012 yang telah bersama-sama berbagi suka duka di jurusan yang sulit ini.
* Teman-teman Eltoro 2012 dan HME ITB yang telah menjadi tempat penulis menghabiskan waktu untuk belajar, bermain, bersosialisasi, dan berorganisasi.
* Dinda Ayu Wirda selaku teman hidup yang selalu memberikan perhatiannya pada penulis.

Penulis menyadari bahwa buku tugas akhir ini tidak sempurna. Oleh sebab itu, penulis sangat mengharapkan kritik dan saran. Semoga laporan dan tugas akhir ini dapat bermanfaat bagi pembaca dan bagi perkembangan ilmu pengetahuan dan teknologi di Indonesia.

Bandung, 16 Juni 2016

Penulis

**DAFTAR ISI**

[ABSTRAK iii](#_Toc451466937)

[ABSTRACT iv](#_Toc451466938)

[PRAKATA v](#_Toc451466939)

[DAFTAR ISI vii](#_Toc451466940)

[DAFTAR GAMBAR ix](#_Toc451466941)

[DAFTAR TABEL xi](#_Toc451466942)

[BAB I PENDAHULUAN 1](#_Toc451466943)

[1.1 Latar Belakang 1](#_Toc451466944)

[1.2 Rumusan Masalah 2](#_Toc451466945)

[1.3 Tujuan 3](#_Toc451466946)

[1.4 Batasan Masalah 3](#_Toc451466947)

[1.5 Metodologi 3](#_Toc451466948)

[1.6 Sistematika Penulisan 4](#_Toc451466949)

[BAB II TINJAUAN PUSTAKA 5](#_Toc451466950)

[2.1 IBM Bluemix 5](#_Toc451466951)

[2.2 IBM Presence Insights 6](#_Toc451466952)

[2.3 IBM Cloudant NoSQL 7](#_Toc451466953)

[2.4 SDK for Node.js 8](#_Toc451466954)

[2.5 JSON 8](#_Toc451466955)

[2.6 MQTT 9](#_Toc451466956)

[2.7 HTTP 9](#_Toc451466957)

[2.8 *Beacon Payload* 10](#_Toc451466958)

[BAB III RANCANGAN DAN IMPLEMENTASI 12](#_Toc451466959)

[3.1 Spesifikasi 12](#_Toc451466960)

[3.2 Rancangan *Front-end* dan *Back-end Application* 13](#_Toc451466961)

[3.3 Implementasi *Front-end* dan *Back-end Application* 15](#_Toc451466962)

[3.3.1 Implementasi Sistem Registrasi 18](#_Toc451466963)

[3.3.2 Implementasi *Beacon Information Request* 22](#_Toc451466964)

[3.3.3 Implementasi *Notification System* 24](#_Toc451466965)

[3.3.4 Implementasi *Attendance Monitoring System* 27](#_Toc451466966)

[3.3.5 Implementasi IBM Presence Insights 30](#_Toc451466967)

[BAB IV PENGUJIAN 39](#_Toc451466968)

[4.1 Pengujian Sistem Registrasi 39](#_Toc451466969)

[4.2 Pengujian *Beacon Information Request* 41](#_Toc451466970)

[4.3 Pengujian *Notification System* 42](#_Toc451466971)

[4.4 Pengujian *Attendance Monitoring System* 43](#_Toc451466972)

[4.5 Pengujian IBM Presence Insights 44](#_Toc451466973)

[BAB V KESIMPULAN DAN SARAN 46](#_Toc451466974)

[5.1 Kesimpulan 46](#_Toc451466975)

[5.2 Saran 46](#_Toc451466976)

[DAFTAR PUSTAKA 47](#_Toc451466977)

**DAFTAR GAMBAR**

[Gambar 2.1 Tampilan Dashboard IBM Bluemix pada menu Catalog. © IBM Corp. 5](#_Toc451467017)

[Gambar 2.2 Tampilan Dashboard IBM Presence Insights. © IBM Corp. 6](#_Toc451467018)

[Gambar 2.3 Tampilan salah satu informasi beacon pada IBM Cloudant NoSQL database. © IBM Corp. 7](#_Toc451467019)

[Gambar 2.4 Logo Node.js. © Node.js Foundation. 8](#_Toc451467020)

[Gambar 2.5 Skema transaksi data pada MQTT. © System Insights. 9](#_Toc451467021)

[Gambar 2.6 Skema transaksi data pada HTTP. © Study CCNA. 10](#_Toc451467022)

[Gambar 3.1 Diagram blok subsistem cloud. © Dokumentasi Penulis. 13](#_Toc451467023)

[Gambar 3.2 Back-end Application dan service-nya, terdiri dari Node.js dan IBM Cloudant NoSQL database. Tampilan ini ada pada konfigurasi IBM Bluemix via web browser. © IBM Corp. 17](#_Toc451467024)

[Gambar 3.3 Diagram alir untuk fungsi Beacon Information Request. © Dokumentasi Penulis. 22](#_Toc451467025)

[Gambar 3.4 Diagram alir untuk fungsi Notification System. © Dokumentasi Penulis. 24](#_Toc451467026)

[Gambar 3.5 Diagram alir untuk fungsi Attendance Monitoring System. © Dokumentasi Penulis. 27](#_Toc451467027)

[Gambar 3.6 IBM Presence Insights service. Tampilan ini ada pada konfigurasi IBM Bluemix via web browser. © IBM Corp. 31](#_Toc451467028)

[Gambar 3.7 Diargam alir penerusan beacon payload ke IBM Presence Insights. © Dokumentasi Penulis. 31](#_Toc451467029)

[Gambar 3.8 Tampilan IBM Presence Insights ketika mengkonfigurasi beacon pada menu management. © Dokumentasi Penulis. 34](#_Toc451467030)

[Gambar 4.1 Informasi pengguna yang tersimpan pada database. © Dokumentasi Penulis. 39](#_Toc451467031)

[Gambar 4.2 Tampilan aplikasi menolak registrasi pengguna menggunakan username yang sama. © Dokumentasi Penulis. 39](#_Toc451467032)

[Gambar 4.3 Tampilan aplikasi menolak sign-in karena password tidak cocok dengan database. © Dokumentasi Penulis. 40](#_Toc451467033)

[Gambar 4.4 Tampilan edit profile untuk mengubah data registrasi pengguna. © Dokumentasi Penulis. 41](#_Toc451467034)

[Gambar 4.5 Informasi beacon yang tersimpan pada database. © Dokumentasi Penulis. 42](#_Toc451467035)

[Gambar 4.6 Aplikasi menampilkan informasi yang sama seperti pada database sesuai dengan beacon yang terdeteksi. © Dokumentasi Penulis. 42](#_Toc451467036)

[Gambar 4.7 Front-end application siap mengirimkan notifikasi ke anggota unit mahasiswa. © Dokumentasi Penulis. 43](#_Toc451467037)

[Gambar 4.8 Notifikasi dengan konten yang sama diterima pengguna. © Dokumentasi Penulis. 43](#_Toc451467038)

[Gambar 4.9 Mode start class pada mobile application diaktifkan pengguna. © Dokumentasi Penulis. 43](#_Toc451467039)

[Gambar 4.10 Daftar nama pengguna yang hadir di kelas muncul pada front-end application. © Dokumentasi Penulis. 44](#_Toc451467040)

[Gambar 4.11 Tampilan grafik jumlah kunjungan pada spot kampus yang telah dipasangi beacon dalam kurun waktu satu minggu. © Dokumentasi Penulis. 44](#_Toc451467041)

[Gambar 4.12 Tampilan persebaran jumlah kunjungan pada jam-jam tertentu dalam kurun waktu satu minggu. © Dokumentasi Penulis. 45](#_Toc451467042)

[Gambar 4.13 Tampilan jumlah kunjungan pada peta lokasi tempat beacon dipasang. © Dokumentasi Penulis. 45](#_Toc451467043)

**DAFTAR TABEL**

[Tabel 3.1 Spesifikasi umum subsistem Front-end dan Back-end Application 12](#_Toc453667250)

[Tabel 3.2 Gambaran umum subsistem Back-end Application. 15](#_Toc453667251)

[Tabel 3.3 Modul Node.js. 17](#_Toc453667252)

[Tabel 3.4 Penggunaan topic pada MQTT broker untuk setiap fungsi registrasi. 18](#_Toc453667253)

[Tabel 3.5 Penggunaan format JSON untuk setiap fungsi registrasi. 19](#_Toc453667254)

[Tabel 3.6 Penggunaan topic pada MQTT broker untuk fungsi beacon information request. 23](#_Toc453667255)

[Tabel 3.7 Penggunaan format JSON untuk fungsi beacon information request. 23](#_Toc453667256)

[Tabel 3.8 Jenis Notifikasi dan Penerimanya. 25](#_Toc453667257)

[Tabel 3.9 Penggunaan format JSON untuk pengiriman notifikasi. 25](#_Toc453667258)

[Tabel 3.10 Detail HTTP POSTS untuk pengiriman data ke PI. 33](#_Toc453667259)

[Tabel 3.11 Sites dan Zones yang telah dikonfigurasikan pada PI. 34](#_Toc453667260)

[Tabel 3.12 State dan penjelasannya dari detail event yang dikirimkan PI. 36](#_Toc453667261)

[Tabel 3.13 Data dan analitik yang disediakan PI 36](#_Toc453667262)

# PENDAHULUAN

* 1. **Latar Belakang**

Banyak tantangan yang dihadapi oleh seorang mahasiswa yang baru memulai sekolah pada kampus baru (Mourning). Aliran informasi mengenai berita kampus, acara-acara penting, dan info mengenai kegiatan perkuliahan dibutuhkan oleh mahasiswa untuk mempermudah tantangan yang dihadapinya di kampus.

Bagi kampus, pengecekan kehadiran mahasiswa di kelas masih menjadi masalah yang sulit untuk diselesaikan. Masalah datang dari kebiasaan mahasiswa melakukan kecurangan dalam pengisian daftar hadir (PT Media Nusantara Citra Tbk, 2015).

Berdasarkan McGraw-Hill Education, 81% mahasiswa perguruan tinggi menggunakan *smartphone* dan tablet untuk belajar pada tahun 2014, 40% meningkat dibandingkan tahun 2013 (Perez). Menurut eMarketer, pada akhir 2015 diperkirakan sekitar 55 juta pengguna *smartphone* di Indonesia. Sedangkan total penetrasi pertumbuhannya mencapai 37,1 persen (Jose, 2015).

*Beacon* adalah perangkat nirkabel yang memancarkan sinyal BLE (*bluetooth low energy*) yang akan ditangkap oleh perangkat komputasi *mobile* (*smartphone*). Aplikasi *smartphone* menerima sinyal BLE dan menyajikan informasi kontekstual berupa deskripsi suatu lokasi dan sekelilingnya (Managing Large Scale, Ultra-Dense Beacon Deployments in Smart Campuses, 2015). Penerapan penggunaan *beacon* yang umum adalah untuk periklanan pada pertokoan retail (Flexible Technologies for Smart Campus, 2016). Sistem uBeacon memanfaatkan keunggulan *beacon* dan menerapkannya pada lingkungan kampus.

Sistem uBeacon merupakan sistem kampus pintar yang dapat digunakan untuk penyebaran informasi berbasis lokasi di kampus, sistem absensi kelas kuliah, dan analisis lalu lintas keramaian di kampus. Sistem ini terdiri dari tiga sub sistem yaitu perangkat iBeacon dengan teknologi BLE (*Bluetooth Low Energy*), aplikasi *smartphone* berbasis platform Android, dan *cloud PaaS* (Platform as a Service) dari IBM Bluemix. Aplikasi ini dirancang untuk mempermudah kehidupan kampus para mahasiswa.

uBeacon merupakan aplikasi *smartphone* yang digunakan untuk menyebarkan berita kampus, acara-acara penting, dan info mengenai kegiatan perkuliahan. uBeacon juga dapat digunakan untuk konfirmasi kehadiran di kelas menggantikan sistem pengisian daftar hadir cara lama. Aplikasi ini dirancang untuk membantu kehidupan kampus para mahasiswa.

uBeacon sebagai aplikasi *smartphone* membutuhkan *back-end application* untuk menangani hal yang tidak bisa dikerjakan sepenuhnya pada perangkat (Google Inc., 2016). *Back-end application* dibangun untuk menangani pemrosesan data dan pertukaran data dengan *database*. *Back-end application* akan dikelola oleh seorang administrator yang tugasnya akan dibantu dengan sebuah *front-end application*. *Front-end application* dibangun untuk membantu tugas administrator dalam mengirimkan notifikasi dan memonitor absensi kelas.

* 1. **Rumusan Masalah**

Masalah yang penulis angkat pada tugas akhir ini adalah sebagai berikut:

* + - 1. cara membangun *front-end* dan *back-end application* yang akan berperan sebagai server dari aplikasi *smartphone* yang akan dibuat;
      2. cara membangun *back-end* yang dapat menyimpan data pengguna;
      3. cara membangun *back-end* yang dapat mengembalikan informasi sesuai *beacon* yang terdeteksi aplikasi;
      4. cara membangun *front-end* yang dapat mengirimkan notifikasi kepada para pengguna aplikasi;
      5. cara membangun *back-end* yang dapat memroses data kehadiran para pengguna aplikasi di kelas kuliah;
      6. cara membangun *front-end* yang dapat menampilkan data kehadiran para pengguna aplikasi di kelas kuliah;
      7. cara menggunakan IBM Presence Insights untuk menampilkan laporan dan analisis keramaian pengguna aplikasi di kampus.
  1. **Tujuan**

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

* + - 1. membangun dan mengimplementasikan *front-end* dan *back-end application* yang dapat melayani kebutuhan aplikasi *smartphone* yang dibuat;
      2. membangun sistem registrasi yang dapat menyimpan data pengguna aplikasi;
      3. membangun *back-end* yang dapat mengirimkan informasi dari *database* sesuai dengan *beacon* yang terdeteksi aplikasi;
      4. membangun *front-end* yang dapat mengirimkan notifikasi pada pengguna aplikasi spesifik;
      5. membangun *back-end* yang dapat memroses data kehadiran pengguna aplikasi di kelas kuliah sesuai dengan *database*;
      6. membangun *front-end* yang dapat menampilkan data kehadiran para pengguna aplikasi di kelas kuliah;
      7. melakukan konfigurasi IBM Presence Insights agar dapat menampilkan laporan dan analisis keramaian berdasarkan data yang dikirimkan pengguna aplikasi di kampus.
  1. **Batasan Masalah**

Batasan masalah yang penulis rumuskan adalah sebagai berikut:

* + - 1. pengembangan dan implementasi aplikasi *smartphone* uBeacon sebagai bagian dari sistem kampus pintar uBeacon;
      2. pendeteksian sinyal *bluetooth* dari *beacon* oleh *smartphone*;
      3. penggunaan dan penempatannya *beacon* pada area kampus;
  1. **Metodologi**

Metode yang digunakan dalam proses pengerjaan tugas akhir ini adalah sebagai berikut:

* + - 1. Penentuan Topik
      2. Pembuatan Proposal Tugas Akhir
      3. Penentuan Spesifikasi Produk
      4. Studi Pustaka
      5. Perancangan *Front-end* dan *Back-end Application*
      6. Implementasi *Front-end* dan *Back-end Application*
      7. Pengujian dan Perbaikan *Front-end* dan *Back-end Application*
      8. Penarikan Kesimpulan dan Pemberian Saran
  1. **Sistematika Penulisan**

Buka tugas akhir ini disusun menjadi 5 buah bab dengan sistematika penulisan sebagai berikut:

* BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika penulisan.

* BAB II TINJAUAN UMUM

Bab ini berisi tentang tinjauan umum tentang perangkat lunak dan teknologi yang akan digunakan dalam pengembangan sistem.

* BAB III PERANCANGAN

Bab ini berisi tentang spesifikasi dan perancangan pengembangan sistem.

* BAB IV IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi proses implementasi dan pengujian dari sistem yang telah dibangun.

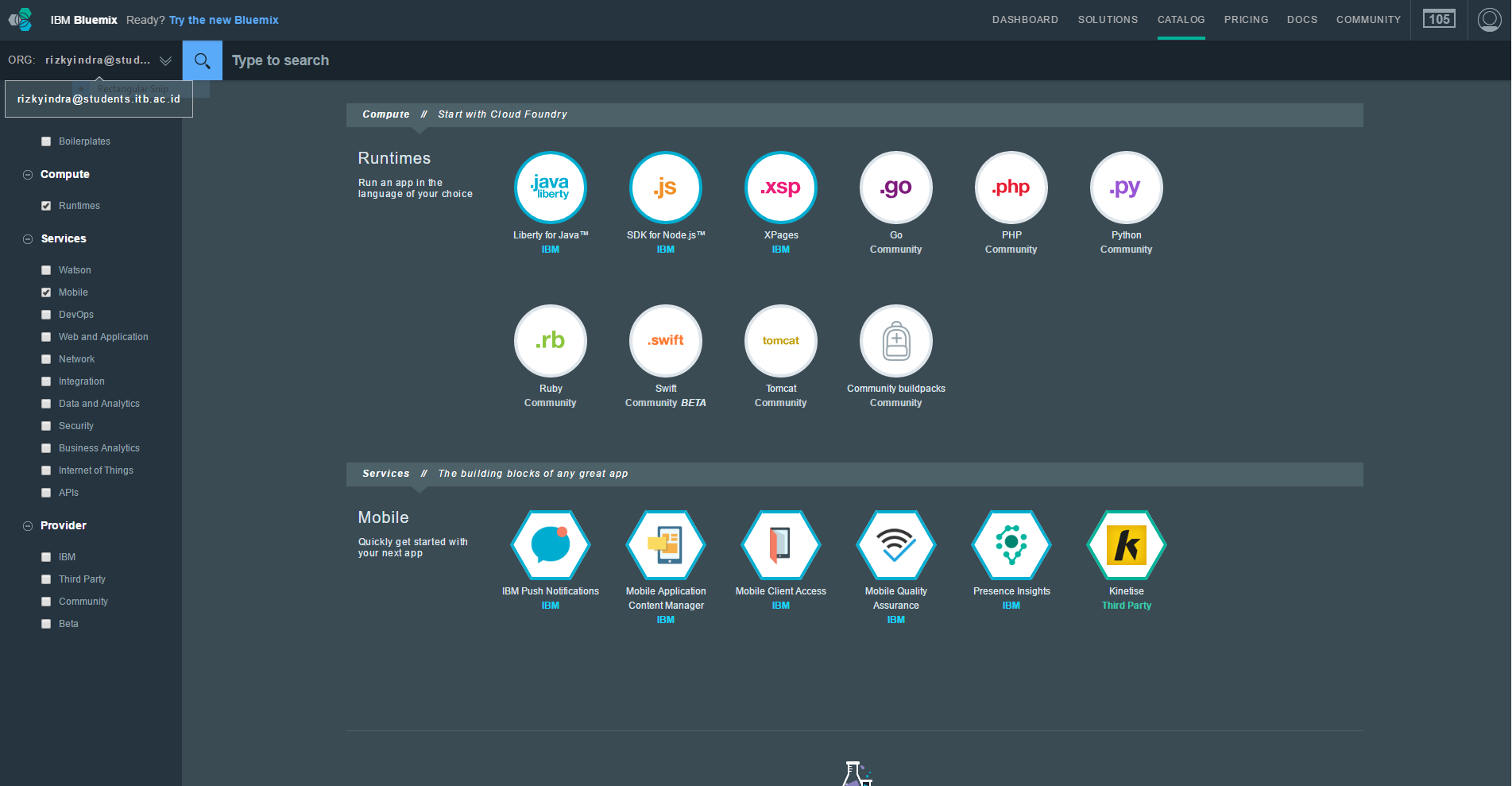
* BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran yang dapat diberikan untuk pengembangan sistem lebih lanjut.

# TINJAUAN PUSTAKA

* 1. **IBM Bluemix**

IBM Bluemix merupakan infrastruktur komputasi awan buatan IBM. IBM Bluemix memudahkan perusahaan dan pengembang untuk membuat *cloud application* dengan mudah. Bluemix merupakan implementasi dari Open Cloud Architecture buatan IBM berbasis Cloud Foundry, sebuah PaaS (*Platform as a Service*) *open source*. IBM Bluemix memiliki *services* yang dapat terintegrasi dengan *cloud application* dengan mudah sehingga tidak memerlukan pengetahuan yang detail tentang cara penginstalan atau konfigurasinya (International Business Machines Corporation, 2015). IBM Bluemix menunjang beberapa bahasa pemrograman dan *runtimes* seperti Java, Node.js, PHP, dan Python.

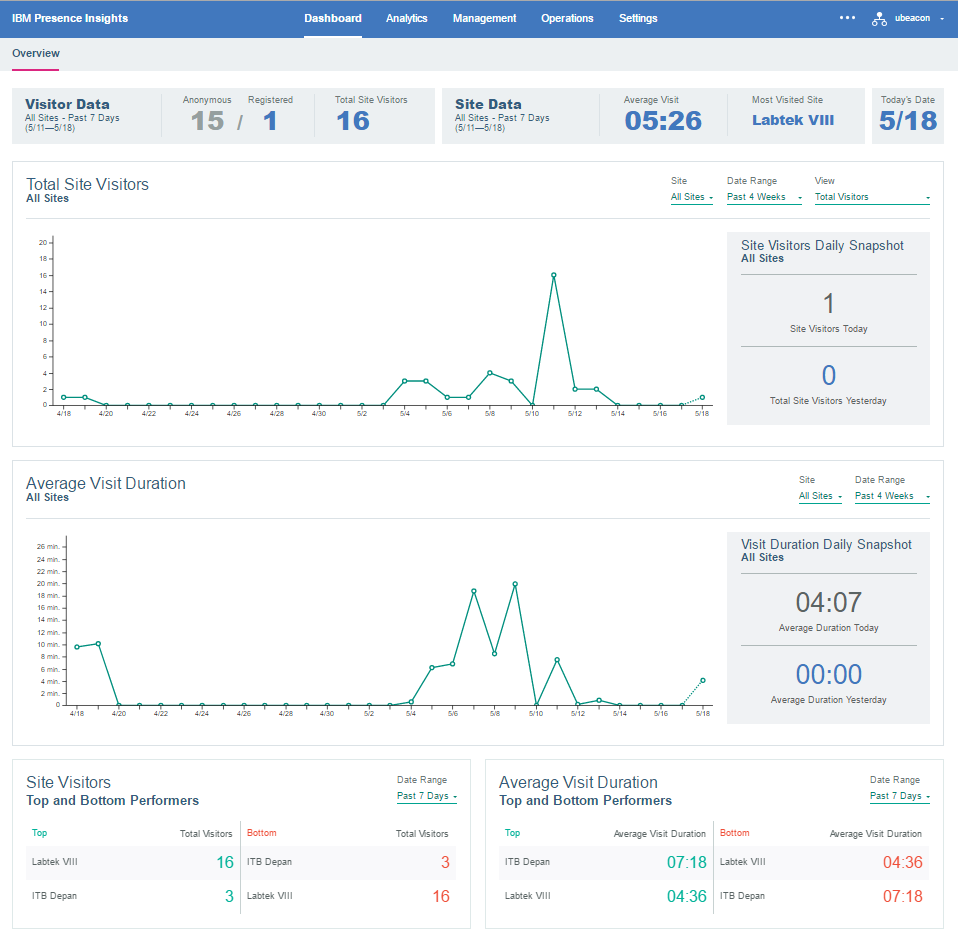


**Gambar 2.1** Tampilan Dashboard IBM Bluemix pada menu Catalog. © IBM Corp.

Pada sistem uBeacon, jasa komputasi awan IBM Bluemix digunakan untuk pengembangan dan pembuatan *back-end application* karena dibutuhkan salah satu dari *services*-nya yang diperlukan untuk analisis. Pengembang juga memiliki akses untuk mendapatkan akun *trial* IBM Bluemix selama 6 bulan sehingga pengembangan dan pembuatan sistem tidak mengeluarkan biaya untuk pembayaran akun.

* 1. **IBM Presence Insights**

IBM Presence Insights (PI) merupakan salah satu *services* yang disediakan oleh IBM Bluemix. PI merupakan sebuah *services* yang telah disediakan dan dikonfigurasikan untuk dapat memberikan laporan analisis mengenai pergerakan pengguna perangkat (*smartphone*) dalam sebuah lokasi.



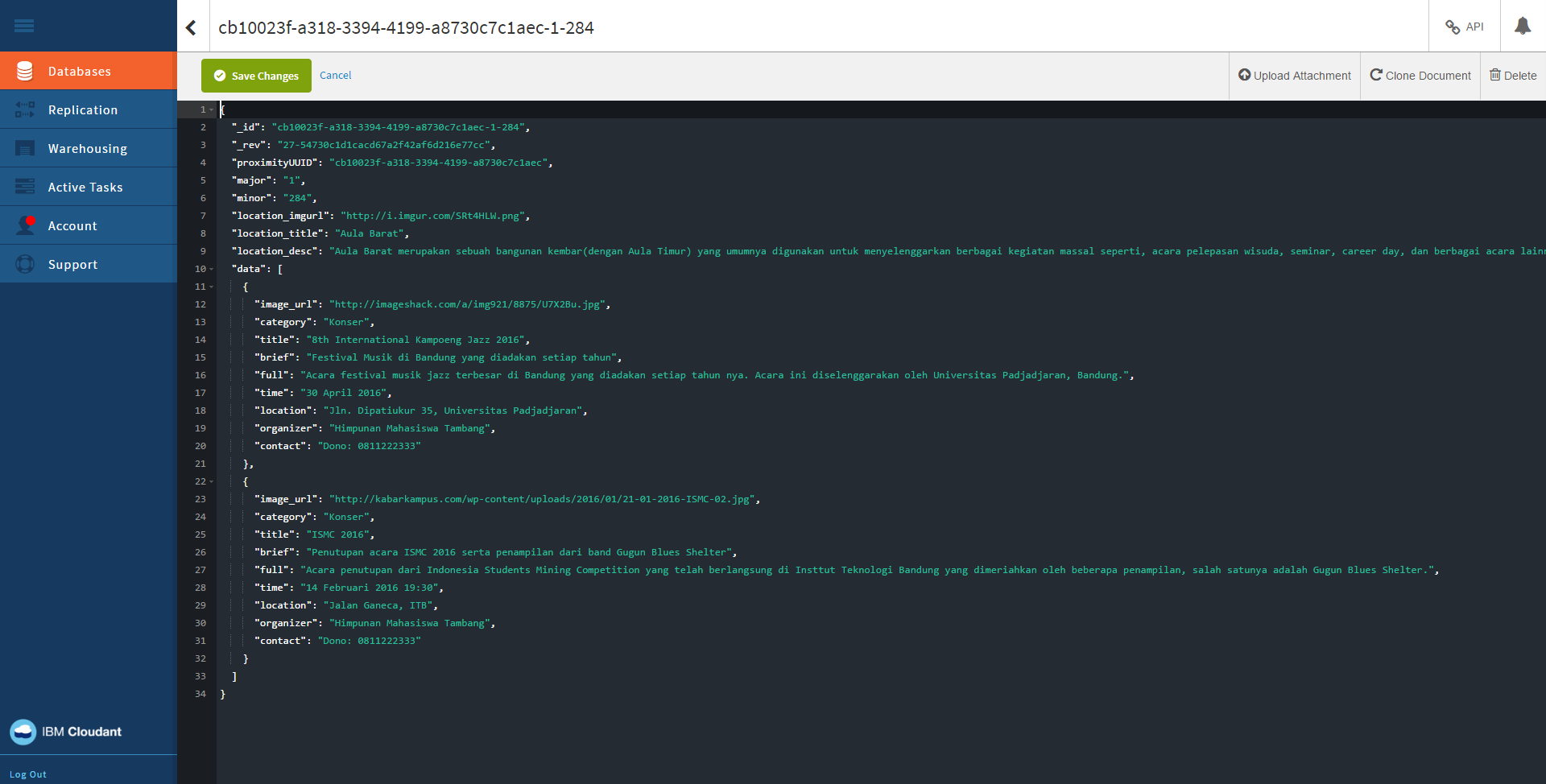
**Gambar 2.2** Tampilan Dashboard IBM Presence Insights. © IBM Corp.

Pergerakan dan perpindahan pengguna *smartphone* dalam beberapa *sites* dan *zones* yang telah dikonfigurasikan sebelumnya dapat dipantau oleh PI. Laporan yang disajikan berupa analisis mendalam mengenai data pengguna *smartphone* seperti pola pergerakan dan lalu lintas pada setiap *sites* (International Business Machines Corporation).

IBM Presence Insights digunakan karena fungsinya yang dapat menghasilkan laporan dan analisis yang instan. Laporan dan analisis yang dihasilkan hanya memerlukan *beacon payload* untuk setiap waktu *smartphone* mendeteksi *beacon*. *Service* ini masih jarang ditemukan pada jasa komputasi awan lain.

* 1. **IBM Cloudant NoSQL**

IBM Cloudant NoSQL digunakan sebagai *database* *service* dari *back-end application*  yang akan dibuat. Semua bentuk data yang perlu disimpan untuk digunakan kembali harus disimpan pada sebuah *database*. IBM Cloudant NoSQL merupakan layanan *database* yang sesuai karena sudah tergabung dalam paket layanan *cloud computing* yang disediakan oleh IBM Bluemix.



**Gambar 2.3** Tampilan salah satu informasi beacon pada IBM Cloudant NoSQL database. © IBM Corp.

IBM Cloudant NoSQL tidak mengimplementasikan SQL pada *database*-nya. Cloudant menyimpan data pada *database* dalam bentuk dokumen-dokumen JSON yang terpisah. Kelebihan menyimpan data dalam bentuk dokumen-dokumen JSON adalah akses yang mudah untuk setiap *key-value* *pair* dari dokumen JSON karena pengambilan *database* langsung dalam satu paket dokumen.

* 1. **SDK for Node.js**



**Gambar 2.4** Logo Node.js. © Node.js Foundation.

SDK (*Software Development Kit*) Node.js merupakan *framework*/platform berbasis JavaScript yang dibangun di atas JavaScript V8 Engine milik Google Chrome. Node.js didesain untuk membangun aplikasi web yang memiliki I/O intensif seperti web video *streaming*, *single-page application*, dan *back-end application*. Node.js bersifat *open source*, gratis, dan bebas digunakan untuk para developer di seluruh dunia (Tutorials Point (I) Pvt. Ltd.).

Node.js digunakan sebagai bahasa pemrograman pembuatan *back-end application* karena memiliki sifat *asynchronous* dan *event driven* (Tutorials Point (I) Pvt. Ltd.). *Asynchronous* berarti *non-blocking* sehingga semua eksekusi perintah dilakukan tanpa menunggu hasil dari eksekusi perintah sebelumnya. *Event driven* berarti eksekusi perintah dimulai ketika sebuah *event* spesifik (HTTP *request* atau MQTT *receive*) terjadi sehingga beberapa fungsi berbeda dapat dimuat dalam satu *source code*. Sifat-sifat ini mendukung *back-end application* uBeacon yang akan berkomunikasi intensif dengan banyak pengguna aplikasi.

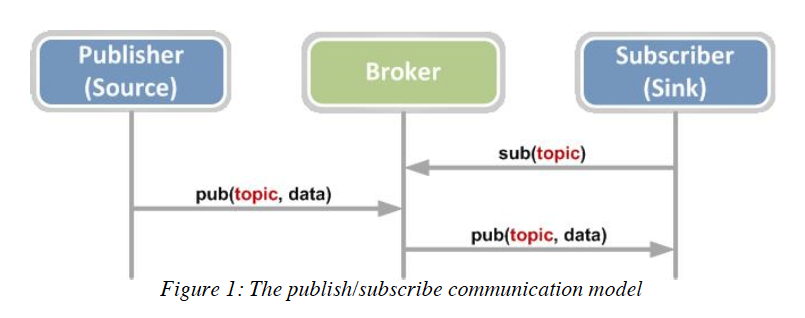
* 1. **JSON**

JSON (*JavaScript Object Notation*) merupakan format standar untuk transmisi data asinkron yang biasa digunakan untuk komunikasi browser/server menggunakan AJAX (Asynchronous JavaScript and XML). JSON berisikan data yang mudah dibaca dengan format pasangan atribut-nilai (*key-value*). JSON merupakan format data yang tidak bergantung pada bahasa pemrograman. JSON diturunkan dari JavaScript, tetapi pada tahun 2016 berbagai bahasa pemrograman telah mendukung fungsi *parsing* JSON (Ecma International).

JSON digunakan sebagai format data umum pada semua komunikasi pada sistem uBeacon. JSON digunakan karena format ini sebenarnya hanya *plain text* yang memiliki format penulisan *key-value*. JSON juga sudah banyak kompatibel dengan berbagai bahasa pemrograman termasuk Node.js. Kesederhanaan format dan kemudahan *parsing*-nya juga membuat JSON dipilih sebagai format data dalam sistem uBeacon.

* 1. **MQTT**

MQTT (*Message Queuing Telemetry Transport*) adalah protokol koneksi *machine-to-machine* yang dirancang untuk transportasi pesan dengan mode *publish*/*subscribe* sehingga pengiriman pesan dapat dilakukan dengan cepat dan ringan. MQTT berguna untuk pengiriman data dengan koneksi terbatas atau *bandwidth* kecil (MQTT Community).



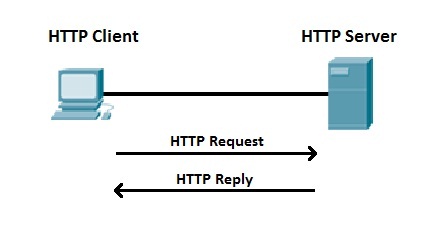
**Gambar 2.5** Skema transaksi data pada MQTT. © System Insights.

Pengiriman pesan pada protokol ini menggunakan pola *publish-subscribe* dengan *publisher* sebagai pengirim pesan dan *subscriber* sebagai penerima pesan. Pola *publish-subscribe* membutuhkan sebuah *broker* yang berperan sebagai distributor pesan ke *client* berdasarkan *topic*-nya. *Topic* berperan sebagai kanal tempat pesan dikirimkan dan diterima oleh *publisher* dan *subscriber*.

MQTT dipilih sebagai komunikasi antara *back-end application* dengan aplikasi *smartphone* uBeacon karena kecepatan dan keringanannya dalam transmisi data. Implementasi MQTT yang mudah pada *back-end application* juga salah satu alasan penggunaan protokol ini.

* 1. **HTTP**

HTTP (*Hypertext Transfer Protokol*) merupakan fondasi dari komunikasi data pada *world wide web*. HTTP merupakan protokol yang digunakan untuk sistem informasi yang terdistribusi dan kolaboratif (R. Fielding, 1999). HTTP berfungsi sebagai protokol *request–response* dalam model komputasi *client–server*.



**Gambar 2.6** Skema transaksi data pada HTTP. © Study CCNA.

Pada HTTP terdapat beberapa metode untuk memberikan instruksi pada server. Beberapa metode di antaranya adalah GET dan POST. GET merupakan metode untuk meminta (*request*) representasi dari alamat *resource* (URL) yang dituju. GET *request* hanya akan menerima data dan tidak melakukan perubahan pada server. POST merupakan metode untuk mengirim sebuah blok data kepada server. POST digunakan untuk melakukan *submit* data yang dibawa di bagian *body* untuk diproses oleh server atau disimpan pada *database*.

Pada sistem uBeacon, HTTP digunakan untuk komunikasi *front-end application* dengan *back-end application*. Pada komunikasi ini HTTP digunakan karena *front-end application* yang dibuat menggunakan JavaScript murni dan tidak memiliki dukungan komunikasi MQTT.

Untuk dapat melakukan komunikasi HTTP, JavaScript pada *front-end application* dapat menggunakan AJAX untuk melakukan GET atau POST ke *back-end application* sehingga pertukaran data dapat dilakukan.

* 1. ***Beacon Payload***

Tidak ada format khusus dalam pertukaran data *beacon* antara aplikasi *smartphone* dan server. Tetapi terdapat standar *payload* yang digunakan dalam pengiriman data *beacon* ke IBM Presence Insights (International Business Machines Corporation). Karena hal ini, maka dalam semua kegiatan pertukaran data dari aplikasi *smartphone* ke server yang melibatkan PI atau tidak akan menggunakan *beacon* *payload* standar ini. Berikut ini adalah format *beacon* *payload* dalam JSON yang digunakan dalam semua aktivitas pertukaran data *beacon*.

|  |
| --- |
| **{**"bnm"**:**  **[{**"descriptor"**:** "rizkyindra"**,**  "detectedTime"**:** 1465300800000**,**  "data"**:{**  "proximityUUID"**:** "cb10023f-a318-3394-4199-a8730c7c1aec"**,**  "major"**:** "1"**,**  "minor"**:** "284"**,**  "rssi"**:** **-**69**,**  "accuracy"**:** 4**,**  "proximity"**:** "Near"  **}**  **}]**  **}** |

# RANCANGAN DAN IMPLEMENTASI

* 1. **Spesifikasi**

Pada proses pengembangannya, beberapa komponen dan teknologi digunakan untuk membangun front-end dan back-end application dari sistem uBeacon. Berikut ini adalah spesifikasi dari komponen dan teknologi yang digunakan.

Tabel 3.1 Spesifikasi umum subsistem Front-end dan Back-end Application

|  |  |  |  |
| --- | --- | --- | --- |
| **Server** | **IBM Bluemix** | * 2 GB Memory * 2 GB Disk Quota * 10 Apps/Services Instances * Sydney, Australia server | |
| **Application** | 1 Instance | * SDK for Node.js (Back-end Application) * HTML, JS, Bootstrap, AJAX (Front-end Application) | |
| **Services** | 2 Instances | **IBM Cloudant NoSQL DB** | * 20 GB Storage |
|  |  | **IBM Presence Insights** | * 500,000 events/month * 1 GB Storage |

Karena pada masa pengembangan akan digunakan akun gratis IBM Bluemix, maka spesifikasi di atas menyesuaikan dengan batasan yang diberikan oleh IBM Bluemix. Agar penggunaan *disk quota* dapat maksimal, maka *disk quota* dengan ukuran 1024 MB hanya akan digunakan untuk data informasi *beacon*, data registrasi pengguna, dan data absensi kelas.

Seorang administrator akan bekerja sebagai bagian dari sistem ini. *Administrator* berperan sebagai orang yang berwenang untuk memperbaharui dan menghapus konten informasi yang ada pada *database*. *Administrator* juga berperan sebagai pengawas dan penjaga seluruh *database* yang ada pada *cloud* *server* mulai dari *database* konten informasi dan *database* registrasi pengguna.

Domain yang digunakan adalah Sydney, Australia karena lokasi yang cukup dekat dengan Indonesia sehingga *latency* yang digunakan untuk mengakses *server* cukup cepat (di bawah 100 ms).

*Instance* adalah jumlah aplikasi yang *running* di IBM Bluemix. Untuk akun gratis, IBM Bluemix menyediakan dua *instances* aplikasi yang dapat berjalan sekaligus pada *server* selama 24 jam. Karena *instance* aplikasi yang diperbolehkan hanya dua, maka keduanya akan dimanfaatkan untuk pembuatan *instance* aplikasi sistem dan *instance* untuk membuat tampilan antarmuka *administrator*.

Karena kerja *server* tidak lebih dari pertukaran informasi *beacon*, penyimpanan registrasi pengguna, dan absensi kelas, maka memori yang dialokasikan untuk kerja *server* hanya 512 MB dari 2048 MB yang disediakan akun gratis IBM Bluemix.

Services & API merupakan paket program yang dapat langsung digunakan untuk kebutuhan analisis data. Untuk keperluan sistem akan dibutuhkan tiga buah *services* yaitu untuk kebutuhan analisis data pengunjung kampus dan untuk *database*.

Untuk akun gratis IBM Bluemix, terdapat batasan HTTP Request yang bisa masuk. Jumlah HTTP Request yang masuk dibatasi <300 request/s. Jumlah ini tidak terlalu membatasi *server* yang akan dibuat karena HTTP Request yang dilakukan oleh *administrator* untuk mengelola *server* tidak akan terlalu banyak.

* 1. **Rancangan *Front-end* dan *Back-end Application***



**Gambar 3.1** Diagram blok subsistem cloud. © Dokumentasi Penulis.

Diagram blok di atas menjelaskan proses pengolahan UUID menjadi konten informasi yang terjadi di *cloud*. Pada basis data, akan didaftarkan UUID dari setiap *beacon* yang aktif bekerja. Setiap UUID yang terdaftar memiliki konten informasi masing-masing. UUID yang dikirim oleh *smartphone* melalui internet akan dicocokkan dengan basis data yang ada. Setelah konten informasi yang sesuai ditemukan, konten informasi akan dikirimkan kembali ke *smartphone* yang mengirimkan UUID untuk selanjutnya diunduh oleh *smartphone*.

*Administrator* bekerja sebagai pengelola *cloud server*. *Administrator* membantu dalam pengelolaan *database* UUID, *database* akun pengguna, konten informasi, dan pemeliharaan *cloud*. *Administrator* berperan sebagai pemegang akses utama untuk menambah, mengubah, dan menghapus semua jenis data yang tersimpan di *cloud*. *Administrator* juga menjadi orang yang harus selalu dapat dihubungi pihak penyelenggara acara agar konten informasi terkait dapat di muat ke dalam basis data.

*Cloud server* yang digunakan bukan merupakan infrastruktur yang dibangun oleh pembuat aplikasi, melainkan menggunakan jasa penyewaan dari pihak ketiga. Dengan menggunakan jasa penyewaan dari pihak ketiga, segala kebutuhan akan keamanan *database* dan perawatan *server* menjadi tanggung jawab penuh penyedia jasa. Kebutuhan untuk menambah tenaga kerja pengelola *server* juga dapat dikurangi. Dengan demikian, fokus perancangan sistem dapat sepenuhnya diberikan pada integrasi ketiga sub-sistem.

Dalam pengoperasian *cloud computing*, pekerjaan *administrator* akan dibantu oleh antarmuka web (*front-end*). *Front-end* membantu *administrator* untuk manajemen *database*, pengiriman notifikasi, dan untuk *monitoring* sistem absensi. Manajemen *database* yang akan dilakukan pada *front-end* dibuat untuk mempermudah pembacaan *database* oleh *administrator*. Untuk pengubahan dan penambahan *database*, *adminsitrator* tetap harus melakukannya manual melalui web IBM Bluemix.

Perangkat lunak yang disediakan jasa *cloud server* akan digunakan untuk segala proses yang dilakukan di *cloud*. Proses-proses tersebut meliputi penerimaan UUID, pencarian konten dan penerimaan kembali informasi, serta proses pengelolaan yang dilakukan oleh *administrator*.

* 1. **Implementasi *Front-end* dan *Back-end Application***

Berikut ini tabel gambaran umum dari sub-sistem *back-end application*.

**Tabel 3.2** Gambaran umum subsistem Back-end Application.

|  |  |
| --- | --- |
| Blok | Fungsi |
| Fungsi | * Mengolah data pengguna (*user*) aplikasi Android * Mengolah data *beacon* dan mengembalikannya menjadi informasi *beacon* yang sesuai * Mengolah data pengguna yang mengirimkan data *beacon* kelas menjadi daftar hadir sistem absensi * Mengirimkan notifikasi kepada pengguna sesuai dengan kegiatannya * Mengolah *traffic* pengguna aplikasi yang melewati daerah-daerah yang telah dipasangi *beacon*. |
| Input | * Data registrasi pengguna * Data beacon yang terdeteksi oleh *bluetooth* pada *smartphone* pengguna * Data beacon yang terdeteksi oleh *bluetooth* pengguna *smartphone* yang terdaftar pada kelas kuliah tertentu * Kiriman notifikasi dari *back-end application* melalui antarmuka *front-end* yang digunakan oleh *administrator* * Informasi *beacon* yang telah dimasukkan ke *database* oleh *administrator* |
| Output | * Data registrasi pengguna yang telah disimpan pada *database* dan dapat diubah sewaktu-waktu oleh pengguna * Informasi tentang *beacon* yang terdeteksi dikirim ke *smartphone* pengguna * Daftar peserta kelas yang hadir di kelas kuliah karena *smartphone*-nya yang mendeteksi *beacon* * Informasi notifikasi pada *smartphone* pengguna * Data dan hasil analisis dari IBM Presence Insights |
| Kebutuhan kuantitatif | * IBM Bluemix sebagai *server* tempat berjalannya *back-end application* dan antarmuka *front-end application* yang digunakan oleh *administrator*. |
| Deskripsi kebutuhan performansi | * Dapat melakukan pengolahan data registrasi pengguna * Dapat melakukan pengolahan data *beacon* dan mengirimkan kembali informasi yang sesuai * Dapat mengolah data *beacon* menjadi daftar hadir sistem absensi kelas * Dapat mengirimkan notifikasi kepada pengguna aplikasi Android * Dapat mengolah *traffic* pengguna aplikasi selama berada di sekitar daerah-daerah yang telah dipasangi *beacon*. |

Sub-sistem *back-end application* menangani semua *request* informasi yang masuk dari aplikasi Android mulai dari sistem *sign-in* sampai dengan permintaan informasi sesuai dengan *beacon* yang terdeteksi oleh *smartphone*. *Back-end application* juga tempat semua data pengguna dan data informasi di simpan. Penyimpanan data tersebut dilakukan pada sebuah *database*.

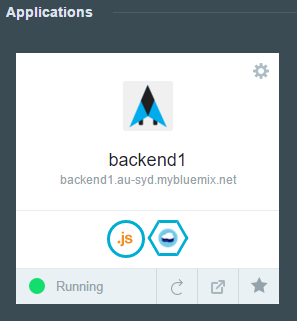
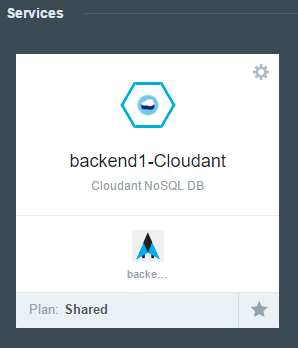
*Front-end* yang dibangun akan digunakan oleh *administrator* sebagai jembatan antara *administrator* dengan *back-end application*. Komunikasi antara *front-end* dengan *back-end* menggunakan AJAX (Asynchronous Javascript and XML). Dengan *front-end*, *administrator* dapat melakukan pengiriman notifikasi tanpa mengubah *source code* program *back-end application*. Daftar hadir sistem absensi yang diolah oleh *back-end application* juga dikirimkan dan ditampilkan ke *front-end* menggunakan AJAX.

Protokol *request* yang digunakan adalah MQTT (*Message Queuing Telemetry Transport*). Melalui protokol ini, sebuah pesan dikirim melalui MQTT *broker*. Di dalam broker tersebut terdapat berbagai nama *topic* yang dapat di-*subscribe* dan *publish*. *Topic* yang digunakan berbeda-beda untuk setiap jenis *request* informasi yang dilakukan.

*Back-end* *application* menerima segala bentuk *request* yang masuk dalam bentuk paket data objek JSON (*JavaScript Object Notation*). JSON yang masuk kemudian di-*parsing* sesuai dengan kebutuhan untuk kemudian diproses lebih lanjut. *Response* balasan dari *back-end application* juga dikirimkan kembali ke aplikasi Android dalam bentuk JSON.

Data yang dikirimkan dari *back-end application* sebagai *response* dari *request* aplikasi Android diambil dari *database*. *Database* menyimpan data registerasi user dan data informasi *beacon* per UUID-major-minor. Ketika pengguna aplikasi Android melakukan *sign-in* ataupun *sign-up*, *back-end application* akan melakukan pencocokan dan penambahan pada database. Informasi *beacon* per UUID-major-minor diatur oleh *administrator*.

Sub-sistem *back-end application* merupakan program yang bekerja pada *server* dan menerima *request* dari aplikasi Android. Tugas utama *back-end application* adalah menerima data *sign-in*/*sign-up* pengguna, menerima data *beacon*, mengembalikan status berhasil/gagalnya *sign-in*/*sign-up*, dan mengembalikan data informasi *beacon* per UUID-major-minor.

**Gambar 3.2** Back-end Application dan service-nya, terdiri dari Node.js dan IBM Cloudant NoSQL database. Tampilan ini ada pada konfigurasi IBM Bluemix via web browser. © IBM Corp.

*Back-end application* dijalankan pada *server* IBM Bluemix dengan Node.js Runtime. Agar *back-end application* dapat menjalankan tugasnya, maka dalam pembuatannya perlu disertakan modul-modul Node.js untuk membantu memudahkan penggunaan fungsi-fungsi tertentu. Berikut ini adalah modul Node.js yang digunakan.

**Tabel 3.3** Modul Node.js.

|  |  |
| --- | --- |
| Node.js Module | Fungsi |
| Express framework | Express framework merupakan modul tambahan untuk implementasi Node.js sebagai *web* dan *mobile application*. Fungsi yang digunakan dari modul ini adalah agar *back-end application* yang dibuat dapat diakses via internet |
| Cloud foundry | Cloud foundry merupakan modul untuk *deploy* program yang telah dibuat ke *server*. Fungsinya adalah untuk menyesuaikan *environment* program dengan *server* tujuan *deploy*. |
| MQTT | MQTT merupakan modul untuk melakukan komunikasi dengan protokol MQTT via internet. Fungsi *publish* dan *subscribe* dilakukan dengan modul ini. |
| Nano | Nano merupakan modul untuk melakukan akses *database*. Pada modul ini terdapat berbagai *syntax* untuk akses *database* (read, write, delete, dll). |
| Cloudant | Cloudant merupakan modul untuk autentikasi akses *database* di IBM Cloudant NoSQL Database. |
| HTTPS | HTTPS digunakan untuk melakukan HTTPS POSTS dari *back-end application* ke IBM Presence Insights |
| Body-Parser | Body-Parser digunakan untuk melakukan *parsing* JSON |

Komunikasi *back-end application* dengan aplikasi Android menggunakan protokol MQTT via internet. Untuk melakukan komunikasi, baik *back-end application* maupun aplikasi Android harus melakukan *publish* dan *subscribe* ke sebuah *topic* pada MQTT Broker. *Topic* yang digunakan harus berbeda-beda agar tidak terjadi miskomunikasi antara *back-end application* dengan aplikasi Android dengan pengguna yang berbeda-beda.

* + 1. Implementasi Sistem Registrasi

Untuk pengiriman data *sign-up* pengguna, dilakukan pengiriman paket data JSON berisi data-data pengguna oleh aplikasi Android. Paket data ini dikirimkan ke *back-end* *application* melalui *topic* “ubeacon/user/signup”. Aplikasi Android melakukan *publish* paket data JSON ke *topic* ini, kemudian *back-end application* melakukan *subscribe* ke *topic* yang sama agar dapat menerima paket data JSON.

Pengiriman kembali status berhasilnya *sign-up* dilakukan oleh *server* ke aplikasi Android. Status dikirimkan melalui *topic* “ubeacon/user/signup/<username>” dengan *username* diambil dari isian pengguna pada saat melakukan *sign-up*. *Topic* yang digunakan harus berbeda agar *back-end application* mengirimkan status kepada *requester*.

Skema yang sama juga digunakan untuk *sign-in*. Paket data JSON berisi *username* dan *password* pengguna dikirimkan dari aplikasi Android ke *back-end application* menggunakan *topic* “ubeacon/user/signin”, kemudian *back-end application* mengembalikan status melalui *topic* “ubeacon/user/signin/<username>”.

Untuk melakukan perubahan data pengguna melalui menu *edit profile* skema yang digunakan sama dengan *sign-up*, hanya saja untuk *edit profile* ada data tambahan berupa *password* lama dan *password* baru. *Topic* yang digunakan untuk mengirim data yang akan diubah adalah ”ubeacon/user/editprofile/”, untuk menerima balasan dari *back-end application* adalah ‘ubeacon/user/editprofile/<username>”.

**Tabel 3.4** Penggunaan topic pada MQTT broker untuk setiap fungsi registrasi.

|  |  |  |
| --- | --- | --- |
| Fungsi | Android app publish to  Back-end subscribe to | Backend publish to  Android app subscribe to |
| *Sign-up* | ubeacon/user/signup | ubeacon/user/signup/<username> |
| *Sign-in* | ubeacon/user/signin | ubeacon/user/signin/<username> |
| *Edit profile* | ubeacon/user/editprofile | ubeacon/user/editprofile/<username> |

**Tabel 3.5** Penggunaan format JSON untuk setiap fungsi registrasi.

|  |  |
| --- | --- |
| Fungsi | JSON Format |
| *Sign-up* | **{**  "\_id"**:** "rizkyindra"**,**  "password"**:** "asdfghjkl;"**,**  "fullname"**:** "Rizky Indra Syafrian"**,**  "email"**:** "rzkyndr@gmail.com"**,**  "birthdate"**:** "1994-08-02"**,**  "occupation"**:** "Mahasiswa"**,**  "interest"**:** "Seminar"  **}** |
| *Sign-in* | **{**  "\_id"**:** "rizkyindra"**,**  "password"**:** "asdfghjkl;"  **}** |
| *Edit profile* | **{**  "\_id"**:** "rizkyindra"**,**  "oldpassword"**:** "asdfghjkl;"**,**  "password"**:** "asdfghjkl"**,**  "fullname"**:** "Rizky Indra Syafrian"**,**  "email"**:** "rzkyndr@gmail.com"**,**  "birthdate"**:** "1994-08-02"**,**  "occupation"**:** "Mahasiswa"**,**  "interest"**:** "Seminar"  **}** |
| *Sign-up* reply status | **{**  "status"**:**"Registeration Success"  **}**  or  **{**  "status"**:**"Username taken"  **}** |
| *Sign-in* reply status | **{**  "status"**:**"Sign-in success"  **}**  or  **{**  "status"**:**"Wrong password"  **}**  or  **{**  "status"**:**"Username not exist"  **}** |
| *Edit profile* reply status | **{**  "status"**:**"Data successfully changed"  **}**  or  **{**  "status"**:**"Wrong password"  **}** |

Ketika menerima data registrasi pengguna, *back-end application* memeriksa *username* yang diregistrasikan pengguna sudah terpakai atau belum. Jika *username* yang diregistrasikan sudah terpakai maka akan ada balasan bahwa *username* sudah dipakai. Jika belum maka registrasi akan sukses.

Berikut ini adalah potongan *source code* untuk bagian *sign-up*.

|  |
| --- |
| // Topic for user data sign-up  **case** 'ubeacon/user/signup'**:**  // Parse incoming message  **var** signup\_1 **=** message**.**toString**();**  **var** signup\_2 **=** JSON**.**parse**(**signup\_1**);**  **var** signup\_3 **=** signup\_2**.**\_id**;**  // Read 'user' database  **var** db\_user **=** cloudant**.**db**.**use**(**'user'**);**  **var** user\_topic\_signup **=** "ubeacon/user/signup/" **+** signup\_3**;**  db\_user**.**get**(**signup\_3**,** **function(**err**,** req**,** res**){**  // Conditional: If username not used, registeration success, otherwise failed  **if** **(!**req**)** **{**  db\_user**.**insert**(**signup\_2**,** **function(**err**,** req**,** res**)** **{**  console**.**log**(**"Registeration success"**);**  **});**  **var** response **=** "{\"status\":\"Registeration success\"}"**;**  client**.**publish**(**user\_topic\_signup**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}** **else** **{**  console**.**log**(**"Username taken"**);**  **var** response **=** "{\"status\":\"Username taken\"}"**;**  client**.**publish**(**user\_topic\_signup**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}**  **});**  **break;** |

Ketika menerima data *sign-in* pengguna, *back-end application* akan memeriksa *username* yang dimasukkan pengguna berada pada *database*. Jika *username* ada pada *database*, maka akan diperiksa kecocokan *password* yang dimasukkan. Jika keduanya benar maka pengguna akan berhasil *sign-in*. Jika pemeriksaan *username* dan/atau *password* gagal, maka akan ada balasan *username* tidak ada atau *password* salah.

Berikut ini adalah potongan *source code* untuk bagian *sign-in*.

|  |
| --- |
| // Topic for user data sign-in  **case** 'ubeacon/user/signin'**:**  // Parse incoming message  **var** signin\_1 **=** message**.**toString**();**  **var** signin\_2 **=** JSON**.**parse**(**signin\_1**);**  **var** signin\_3 **=** signin\_2**.**\_id**;**  **var** signin\_4 **=** signin\_2**.password;**  // Read 'user' database  **var** db\_user **=** cloudant**.**db**.**use**(**'user'**);**  **var** user\_topic\_signin **=** "ubeacon/user/signin/" **+** signin\_3**;**  db\_user**.**get**(**signin\_3**,** **function(**err**,** req**,** res**){**  // Conditional: If username and password match success, otherwise failed  **if** **(**req**)** **{**  **if** **(**req**.password** **===** signin\_4**)** **{**  console**.**log**(**"Sign-in success"**);**  **var** test **=** req**;**  test**.status** **=** 'Sign-in success'**;**  **var** response **=** JSON**.**stringify**(**test**);**  client**.**publish**(**user\_topic\_signin**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}** **else** **{**  console**.**log**(**"Wrong password"**);**  **var** response **=** "{\"status\":\"Wrong password\"}"**;**  client**.**publish**(**user\_topic\_signin**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}**  **}** **else** **{**  console**.**log**(**"Username not exist"**);**  **var** response **=** "{\"status\":\"Username not exist\"}"**;**  client**.**publish**(**user\_topic\_signin**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}**  **});**  **break;** |

Ketika pengguna mengubah datanya melalui menu *edit profile* pada aplikasi Android, data baru akan dikirimkan ke *back-end application*. Data baru ini berisikan data tambahan berupa *password* lama dan *password* baru. *Password* lama akan dicocokkan, apabila sesuai maka password lama akan dihilangkan dan data-data lain yang mengikut akan diubah. Apabila *password* salah, perubahan data tidak bisa dilakukan.

Berikut ini adalah potongan *source code* untuk melakukan *edit profile*.

|  |
| --- |
| // Topic for edit user data  **case** 'ubeacon/user/editprofile'**:**  // Parse incoming message  **var** edit\_1 **=** message**.**toString**();**  **var** edit\_2 **=** JSON**.**parse**(**edit\_1**);**  **var** edit\_3 **=** edit\_2**.**\_id**;**  **var** edit\_4 **=** edit\_2**.**oldpassword**;**  // Read 'user' database  **var** db\_user **=** cloudant**.**db**.**use**(**'user'**);**  **var** user\_topic\_edit **=** "ubeacon/user/editprofile/" **+** edit\_3**;**  db\_user**.**get**(**edit\_3**,** **function(**err**,** req**,** res**){**  // Conditional: If username and password match success, otherwise failed. Password unchanged if new password value is null;  **if** **(**req**)** **{**  **if** **(**req**.password** **===** edit\_4**)** **{**  **if** **(**edit\_2**.password** **==** ""**)** **{**  edit\_2**.password** **=** edit\_2**.**oldpassword**;**  **}**  **delete** edit\_2**.**oldpassword**;**  edit\_2**.**\_rev **=** req**.**\_rev**;**  db\_user**.**insert**(**edit\_2**,** **function(**err**,** req**,** res**)** **{**  console**.**log**(**"Data successfully changed"**);**  **});**  edit\_2**.status** **=** "Data successfully changed"**;**  **var** doc **=** JSON**.**stringify**(**edit\_2**);**  client**.**publish**(**user\_topic\_edit**,** doc**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}** **else** **{**  console**.**log**(**"Authectication Failed"**);**  **var** response **=** "{\"status\":\"Wrong password\"}"**;**  client**.**publish**(**user\_topic\_edit**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}**  **}** **else** **{**  console**.**log**(**"Username not exist"**);**  **var** response **=** "{\"status\":\"Username not exist\"}"**;**  client**.**publish**(**user\_topic\_edit**,** response**,** **function(){**  console**.**log**(**"Response sent"**);**  **});**  **}**  **});**  **break;** |

* + 1. Implementasi *Beacon Information Request*



**Gambar 3.3** Diagram alir untuk fungsi Beacon Information Request. © Dokumentasi Penulis.

*Beacon Information Request* merupakan prosedur yang dilakukan oleh *back-end application* untuk menerima *payload* dari *mobile application* dan membalasnya dengan informasi yang sesuai dengan UUID *beacon* pada *database*. Penerimaan *payload* dilakukan *back-end application* dengan melakukan *subscribe* pada *topic* “ubeacon/user/request”. Pada *payload* terdapat *username* pengirim, UUID, major, minor, dan data-data lain. Data-data tersebut di-*parsing* untuk diambil UUID, major, dan minor-nya dan kemudian disatukan menjadi satu *string* UUID-major-minor.

Pada *database* dilakukan pencarian *string* yang sesuai untuk mendapatkan informasi yang tersimpan. Setelah ditemukan, informasi diambil dan dikirimkan kembali ke *mobile application* yang mengirimkan *payload*. Pengiriman kembali ini dilakukan *back-end application* dengan *publish* ke *topic* “ubeacon/user/response/<username>” dengan <username> diisi dengan *username* pengirim *payload*.

Selain diproses untuk pertukaran data informasi *beacon*, *payload* yang diterima oleh *back-end application* juga diteruskan ke IBM Presence Insights untuk dimasukkan sebagai bagian dari data analitik lalu lintas kampus.

**Tabel 3.6** Penggunaan topic pada MQTT broker untuk fungsi beacon information request.

|  |  |  |  |
| --- | --- | --- | --- |
| Fungsi | Android app publish to  Back-end subscribe to | Backend publish to  Android app subscribe to | |
| Data dan informasi *beacon* | ubeacon/request | | ubeacon/user/response/<username> |

**Tabel 3.7** Penggunaan format JSON untuk fungsi beacon information request.

|  |  |
| --- | --- |
| Fungsi | JSON Format |
| *Beacon* data | **{**  "bnm"**:** **[**  **{**  "descriptor"**:** "aa:bb:cc:dd:ee:ff"**,**  "detectedTime"**:** "2016-02-02T11:00:00.000Z"**,**  "data"**:** **{**  "proximityUUID"**:** "cb10023f-a318-3394-4199-a8730c7c1aec"**,**  "major"**:** "284"**,**  "minor"**:** "1"**,**  "rssi"**:** 69**,**  "accuracy"**:** 75**,**  "proximity"**:** "Immediate"  **}**  **}**  **]**  **}** |
| *Beacon* information | **{**  "\_id"**:** "cb10023f-a318-3394-4199-a8730c7c1aec-284-1"**,**  "\_rev"**:** "8-f0c5ec74cbc17c2e69aa74b40405f322"**,**  "proximityUUID"**:** "cb10023f-a318-3394-4199-a8730c7c1aec"**,**  "major"**:** "284"**,**  "minor"**:** "1"**,**  "image\_url"**:** "http://backend1.au-syd.mybluemix.net/images/sampleimage1.jpg"**,**  "category"**:** "Sample category 1"**,**  "brief"**:** "This is sample brief description for event 1"**,**  "full"**:** "This is sample long description of the event 1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer feugiat dapibus euismod. Aliquam ac dui non quam ornare tincidunt et consectetur nunc."**,**  "time"**:** "2016-05-01T08:00:00.000Z"**,**  "location"**:** "Sample location 1"**,**  "organizer"**:** "Sample organization 1"**,**  "contact"**:** "Sample contact 1: 081234567890"  **}** |

* + 1. Implementasi *Notification System*



**Gambar 3.4** Diagram alir untuk fungsi Notification System. © Dokumentasi Penulis.

*Notification System* merupakan sistem yang dibuat pada *front-end application* dan *back-end application* untuk mengirimkan isian notifikasi pada *front-end* kepada pengguna spesifik. Pemohon pengirim notifikasi berasal dari lembaga yang memiliki kepentingan memberikan pesan pada anggota lembaganya, contohnya seperti unit kegiatan mahasiswa atau mahasiswa program studi tertentu. Pemohon melakukan *request* pengiriman notifikasi pada administrator. Administrator akan mengirimkan notifikasi melalui *front-end* yang telah dibuat. Ketika tombol *send* pada *front-end* ditekan oleh administrator, semua isian pada *front-end* dipaketkan menjadi sebuah JSON. Paket data dikirimkan ke *back-end application* menggunakan AJAX melalui protokol HTTP dengan metode POST. Paket data JSON dibawa pada *body* HTTP POST.

Notifikasi yang dikirimkan oleh *administrator* akan dibedakan menjadi tiga tipe. Tiga tipe tersebut adalah notifikasi berdasarkan jurusan/program studi, notifikasi berdasarkan unit mahasiswa, dan notifikasi untuk semua pengguna. Setiap notifikasi memiliki *route* HTTP POST dan *topic* sendiri agar notifikasi dapat sampai ke pengguna spesifik. Meskipun notifikasi dibedakan menjadi tiga tipe, prosedur pengirimannya dari *front-end* ke *back-end* tetap sama.

**Tabel 3.8** Jenis Notifikasi dan Penerimanya.

|  |  |  |
| --- | --- | --- |
| Jenis Notifikasi | Penerima Notifikasi | Keterangan |
| *Advertisement* | Semua pengguna |  |
| Unit Mahasiswa | Pengguna anggota unit mahasiswa tertentu | Data keanggotaan unit didapatkan dari data registrasi pengguna |
| Program Studi | Pengguna anggota program studi tertentu | Data program studi dari data NIM registrasi pengguna |

**Tabel 3.9** Penggunaan format JSON untuk pengiriman notifikasi.

|  |  |
| --- | --- |
| Notification | JSON Format |
| *Major member* | **{**  "major"**:**"132"**,**  "batch"**:**"12",  [core body]  **}** |
| *Unit member* | **{**  "unit"**:**"UBG",  [core body]  **}** |
| *Notification core body* | **{**  "image\_url"**:**"Test"**,**  "title"**:**"Test"**,**  "notifications"**:**"Test"**,**  "time"**:**"test"**,**  "location"**:**"Test"**,**  "contact"**:**"Test"  **}** |

Berikut ini adalah *source code* pengiriman data dari *front-end* ke *back-end* yang berupa Javascript berikut dengan *form* notifikasinya dalam bentuk HTML.

Fungsi *submitForm* merupakan pengambilan data dari form HTML yang diisi. Fungsi *postData* merupakan pengiriman data ke *back-end*.

|  |
| --- |
| <html>  <head>  <script type=**"text/javascript"**>  ***function*** submitForm**(){**  ***var*** title **=** document.advertise.title.value**;**  ***var*** notifications **=** document.advertise.notifications.value**;**  ***var*** time **=** document.advertise.time.value**;**  ***var*** location **=** document.advertise.location.value**;**  ***var*** contact **=** document.advertise.contact.value**;**  ***var*** json **=** '{\"title\":\"'**+**title**+**'\",\"notifications\":\"'**+**notifications**+**'\",\"time\":\"'**+**time**+**'\",\"location\":\"'**+**location**+**'\",\"contact\":\"'**+**contact**+**'\"}'  postData**(**json**);**  **}**  ***function*** postData**(**input**){**  ***var*** xhr **=** ***new*** XMLHttpRequest**();**  ***var*** url **=** "/post/advertise"**;**  xhr.open**(**"POST"**,** url**,** ***true*);**  xhr.setRequestHeader**(**'Content-Type'**,** 'application/json'**);**  xhr.send**(**input**);**  alert**(**"Advertisement has been sent"**);**  **}**  </script>  </head> |

|  |
| --- |
| // POST notification to back-end, then MQTT publish to mobile apps  app**.**post**(**'/post/notification/unit'**,** **function** **(**req**,** res**){**  **var** notificationData **=** req**.**body**;**  **var** unitData **=** req**.**body**.**unit**;**  **var** notification\_unit\_topic **=** 'ubeacon/user/notification/unit/' **+** unitData**;**  **delete** notificationData**.**unit**;**  **var** doc **=** JSON**.**stringify**(**notificationData**);**  console**.**log**(**notification\_unit\_topic**);**  client**.**publish**(**notification\_unit\_topic**,** doc**,** **function(){**  console**.**log**(**"JSON sent (unit notification)"**);**  **});**  **});**  // POST notification to back-end, then MQTT publish to mobile apps  app**.**post**(**'/post/notification/major'**,** **function** **(**req**,** res**){**  **var** notificationData **=** req**.**body**;**  **var** majorData **=** req**.**body**.**major**;**  **var** batchData **=** req**.**body**.**batch**;**  **var** notification\_major\_topic **=** 'ubeacon/user/notification/major/' **+** majorData **+** batchData**;**  **delete** notificationData**.**major**;**  **delete** notificationData**.**batch**;**  **var** doc **=** JSON**.**stringify**(**notificationData**);**  client**.**publish**(**notification\_major\_topic**,** doc**,** **function(){**  console**.**log**(**"JSON sent (major notification)"**);**  **});**  **});**  //POST notification to back-end, then MQTT publish to mobile apps  app**.**post**(**'/post/advertise'**,** **function** **(**req**,** res**){**  **var** notificationData **=** req**.**body**;**  **var** doc **=** JSON**.**stringify**(**notificationData**);**  client**.**publish**(**"ubeacon/user/notification/advertise"**,** doc**,** **function(){**  console**.**log**(**"JSON sent (advertise)"**);**  **});**  **});** |

*Source code* di atas merupakan penerusan notifikasi yang diterima dari *front-end* ke pengguna aplikasi melalui MQTT. Notifikasi dari *front-end* dikirimkan melalui HTTP POSTS, pada *back-end* diterima melalui masing-masing *route*-nya dan dikirimkan ke masing-masing *topic* agar sampai ke pengguna spesifik.

Pada *back-end application*, paket data JSON diterima dan kemudian diteruskan ke *mobile application* melalui protokol MQTT. Pada proses ini terjadi perubahan protokol pada transmisi data dari MQTT menjadi HTTP. Hal ini dilakukan karena transmisi data melalui internet dengan protokol MQTT lebih cepat dan lebih ringan (MQTT Community). Selain itu, platform Android tidak lagi menunjang transaksi data melalui protokol HTTP untuk versi terbarunya (Google Inc.).

Paket data JSON yang berisi JSON diteruskan ke *mobile application* dengan melakukan *publish* melalui *topic* “ubeacon/user/notification/<notification type>/<specifier>” dengan <notification type> adalah tipe notifikasi dan <specifier> adalah target spesifik penerima notifikasi.

* + 1. Implementasi *Attendance Monitoring System*



**Gambar 3.5** Diagram alir untuk fungsi Attendance Monitoring System. © Dokumentasi Penulis.

*Attendance Monitoring System* merupakan sistem yang dibuat pada *front-end application* dan *back-end* application untuk memonitor kehadiran mahasiswa di kelas dan menampilkan daftar kehadirannya pada *front-end*. Prosedur yang sama dilakukan sama mirip seperti *Beacon Information Request* yaitu dengan menerima *payload* dari *mobile application*, tetapi *topic* tempat *back-end application* melakukan *subscribe* adalah “ubeacon/user/presence”. Pada *payload* terdapat *username* pengirim, UUID, major, minor, dan waktu pengiriman *payload*. Data-data tersebut di-*parsing* seperti prosedur *Beacon Information Request*.

Berikut ini adalah potongan *source code* yang dibuat untuk sistem absensi.

|  |
| --- |
| **case** 'ubeacon/user/presence'**:**  // Parse incoming message  **var** presence\_1 **=** message**.**toString**();**  **var** presence\_2 **=** JSON**.**parse**(**presence\_1**);**  **var** descriptor **=** presence\_2**.**bnm**[**0**].**descriptor**;**  **var** detectedTime **=** presence\_2**.**bnm**[**0**].**detectedTime**;**  **var** proximityUUID **=** presence\_2**.**bnm**[**0**].**data**.**proximityUUID**;**  **var** majorUUID **=** presence\_2**.**bnm**[**0**].**data**.**major**;**  **var** minorUUID **=** presence\_2**.**bnm**[**0**].**data**.**minor**;**  **var** UUID **=** **(**proximityUUID **+** '-' **+** majorUUID **+** '-' **+** minorUUID**);**  **var** proximity **=** presence\_2**.**bnm**[**0**].**data**.**proximity**;**  // Read all database  **var** db\_courses **=** cloudant**.**db**.**use**(**'courses'**);**  **var** db\_user **=** cloudant**.**db**.**use**(**'user'**);**  **var** db\_beacon **=** cloudant**.**db**.**use**(**'beacon'**);**  // Function to check class attendance name, to be used with findIndex method  **function** checkName**(**object**)** **{**  **return** object**.**name **===** descriptor**;**  **}**  **function** checkId**(**object**)** **{**  **return** object**.**\_id **===** descriptor**;**  **}**  db\_beacon**.**get**(**UUID**,** **function(**err**,** req**,** res**){**  // Get course name  **var** course **=** req**.**course**;**  // Proximity check  **if** **(**proximity **===** 'Near' **||** proximity **===** 'Immediate'**){**  db\_courses**.**get**(**course**,** **function(**err**,** req**,** res**){**  // Switch by course  **switch(**course**){**    **case** 'EL2001-01'**:**  **var** temp1 **=** req**.**students**;**  **var** result **=** temp1**.**findIndex**(**checkName**);**  **if(**result**>=**0**){**  **var** temp2 **=** req**.**students**[**result**];**  temp2**.**timestamp **=** detectedTime**;**  **var** result2 **=** att\_EL2001\_01**.**attendance**.**findIndex**(**checkName**);**  **if(**result2**>=**0**){**  att\_EL2001\_01**.**attendance**[**result2**]** **=** temp2**;**  **}** **else** **{**  att\_EL2001\_01**.**attendance**.**push**(**temp2**);**  **}**  att\_EL2001\_01**.**attendance**.**sort**(**compareNim**);**  console**.**log**(**att\_EL2001\_01**);**  **}**  **if(**descriptor**===**req**.**lecturer**){**  att\_EL2001\_01**.**isLecturerPresent **=** **true;**  **}**  **break;** |

Pada *database* *beacon*, selain informasi *beacon* terdapat juga kode kuliah dan nomor ruangan lokasi *beacon* tersebut ditempatkan. UUID-major-minor dari *payload* dijadikan satu *string* dan dilakukan pencarian pada *database* informasi untuk mengetahui kode kuliah dan nomor ruangan dari *beacon* tersebut.

Setelah diketahui kode kuliah dan nomor ruangan kelas, selanjutnya adalah mengecek *username* pada *database* kelas kuliah untuk mengetahui bahwa pengguna tersebut terdaftar sebagai peserta kelas kuliah. Jika pengguna tidak terdaftar sebagai peserta kelas, maka *payload* yang sudah masuk akan diabaikan. Jika pengguna terdaftar sebagai peserta kelas, maka data *username* dan waktu pengiriman *payload* akan dimasukkan pada sebuah *array*.

*Array* ini merupakan daftar pengguna yang terdaftar dan hadir pada kelas kuliah. *Array* yang sudah berisi *username* dan waktu pengiriman *payload* ini siap untuk diakses dan ditampilkan oleh *front-end application*. *Front-end application* melakukan HTTP GET menggunakan AJAX ke *back-end application* untuk mendapatkan *array* ini. *Array* yang sudah didapatkan kemudian di-*parsing* dan ditampilkan pada *front-end application* untuk dimonitor oleh administrator.

Pada *array* terdapat *username* dan waktu pengiriman *payload*. Waktu pengiriman *payload* digunakan untuk mengetahui waktu terakhir pengguna berada di kelas. Mobile *application* telah dikonfigurasi untuk mengirimkan *payload* setiap 5 menit pada saat pengguna hadir di kelas. *Back-end application* juga melakukan pengecekan selisih waktu terakhir pengguna berada di kelas dengan waktu terkini. Apabila waktu terakhir pengguna berada dikelas lebih besar dari 5 menit, maka pengguna akan dihapuskan dari *array*. Pengecekan *array* ini dilakukan setiap 5 detik.

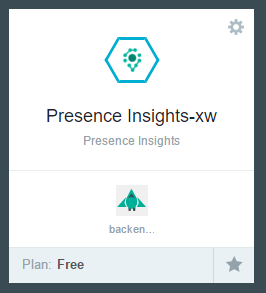
|  |
| --- |
| **var** threshold **=** 300000**;**  **var** routine **=** 5000**;**  **setInterval(function()** **{**  console**.**log**(**"5 seconds check"**);**  **var** d **=** **new** Date**();**  **var** currenttime **=** d**.**getTime**();**  console**.**log**(**currenttime**);**  **var** att2\_EL2001\_01 **=** att\_EL2001\_01**.**attendance**;**  console**.**log**(**"EL2001:"**,** att2\_EL2001\_01**);**  **var** result3 **=** att2\_EL2001\_01**.**findIndex**(**checkTime**);**  **if(**result3**>=**0**){**  att\_EL2001\_01**.**attendance**.**remove**(**result3**);**  result3 **=** **-**1**;**  **}**  **},**routine**);** |

*Front-end application* juga telah dikonfigurasi untuk melakukan *refresh* setiap 5 detik sehingga daftar hadir peserta kelas yang ditampilkan akan selalu *update*. Hal ini berguna untuk memastikan bahwa peserta kuliah yang meninggalkan kelas dapat dihapus dari *array* dan tidak ditampilkan pada *front-end application*.

|  |
| --- |
| <script type=**"text/javascript"**>  ***function*** loadDoc**(){**  ***var*** xhttp **=** ***new*** XMLHttpRequest**();**  xhttp.onreadystatechange **=** ***function*()** **{**  ***if*** **(**xhttp.readyState **==** 4 **&&** xhttp.status **==** 200**)** **{**  ***var*** response1 **=** JSON.parse**(**xhttp.responseText**);**  document.getElementById**(**"course"**).**innerHTML **=** response1.course**;**  document.getElementById**(**"coursename"**).**innerHTML **=** response1.coursename**;**  ***for*(*var*** i**=**0**;** i**<=**response1.attendance.length**;** i**++){**  document.getElementById**(**"name"**+**i**).**innerHTML **=** response1.attendance**[**i**].**name**;**  document.getElementById**(**"nim"**+**i**).**innerHTML **=** response1.attendance**[**i**].**nim**;**  **}**  **}**  **};**  xhttp.open**(**"GET"**,** "/presence/EL2001-01"**,** ***false*);**  xhttp.send**();**  **}**    ***function*** delDoc**(){**  ***for*(*var*** i**=**0**;** i**<=**20**;** i**++){**  document.getElementById**(**"name"**+**i**).**innerHTML **=** " "**;**  document.getElementById**(**"nim"**+**i**).**innerHTML **=** " "**;**  **}**  **}**    setInterval**(*function*()** **{**  delDoc**();**  loadDoc**();**  **},**5000**);**  </script> |

* + 1. Implementasi IBM Presence Insights

IBM Presence Insight merupakan sebuah *service* dari IBM Bluemix yang digunakan untuk melihat data dan analitik dari perangkat yang dapat mendeteksi *beacon*. *Service* ini dapat ditambahkan melalui katalog IBM Bluemix dan dapat di-*binding* dengan *back-end application* yang sudah ada.



**Gambar 3.6** IBM Presence Insights service. Tampilan ini ada pada konfigurasi IBM Bluemix via web browser. © IBM Corp.

IBM Presence Insight (PI) digunakan untuk menerima kiriman paket data JSON dari aplikasi Android yang mendeteksi adanya *beacon* di sekitar *smartphone*. Paket data JSON yang dikirimkan berisi data mengenai *beacon* yang terdeteksi dan waktu terjadinya deteksi. Dari paket data JSON yang diterima ini, dapat dibuat data mengenai jumlah perangkat yang sedang berada di sekitar *beacon*, lamanya perangkat tersebut berada di sekitar *beacon*, para pengguna yang sedang berada di sekitar *beacon*, dan data lainnya yang dapat dikemas dalam sebuah statistik yang berguna untuk analisis pengunjung.



**Gambar 3.7** Diargam alir penerusan beacon payload ke IBM Presence Insights. © Dokumentasi Penulis.

IBM Presence Insight sebenarnya digunakan sebagai penerima pertama data *beacon* yang dikirimkan oleh aplikasi Android. Setelah itu, data *beacon* yang masuk ke PI kemudian diteruskan ke *subscriber*. *Subscriber* merupakan *back-end application* yang menerima data terusan dari PI untuk kemudian diolah lebih lanjut.

Pada implementasi ini, protokol yang digunakan untuk pengiriman data dari aplikasi Android ke *server* adalah MQTT, sedangkan API (*Application Programming Interface*) PI sendiri hanya dapat menerima data melalui protokol HTTP. Untuk itu, skema pengiriman data ke PI perlu diubah. Pengiriman paket data JSON yang berisi data *beacon* akan dikirimkan oleh *back-end application*. Pada *back-end application*, paket data JSON yang masuk melalui protokol MQTT dikirimkan ulang ke PI dengan protokol HTTP.

Paket data JSON yang berisi data *beacon* dengan format standar dikirimkan ke PI melalui API-nya (*Application Programming Interface*). Pengiriman data *beacon* ke PI dilakukan dengan HTTP POSTS ke API *beacon connector* milik PI. Berikut ini adalah detail HTTP POSTS yang dikirimkan ke PI.

|  |
| --- |
| // Function to forward registeration data to IBM Presence Insight  **function** ToPresenceInsight\_RegisterDevices**(**jsoninput\_registerdevices**)** **{**  **var** name **=** jsoninput\_registerdevices**.**fullname**;**  **var** descriptor **=** jsoninput\_registerdevices**.**\_id**;**  **var** registered **=** **true;**  **var** registrationType **=** "Auto"**;**  **var** email **=** jsoninput\_registerdevices**.**email**;**  **var** birthdate **=** jsoninput\_registerdevices**.**birthdate**;**  **var** occupation **=** jsoninput\_registerdevices**.**occupation**;**  **var** interest **=** jsoninput\_registerdevices**.**interest**;**  **var** jsonoutput **=** **(**'{\"name\":\"'**+**name**+**'\",\"descriptor\":\"'**+**descriptor**+**'\",\"registered\":'**+true+**',\"registrationType\":\"Auto\",\"unencryptedData\":{\"email\":\"'**+**email**+**'\",\"birthdate\":\"'**+**birthdate**+**'\",\"occupation\":\"'**+**occupation**+**'\",\"interest\":\"'**+**interest**+**'\"}}'**);**  **var** options **=** **{**  hostname**:** 'presenceinsights.au-syd.bluemix.net'**,**  port**:** 443**,**  path**:** '/pi-config/v1/tenants/1g4a05cq/orgs/my1u08tp/devices'**,**  method**:** 'POST'**,**  headers**:** **{**  'Content-Type'**:** 'application/json'**,**  'Authorization'**:** 'Basic NTJnMDBrbTpiNXhQdzZKLW5USnE='  **}**  **};**  **var** req **=** https**.**request**(**options**,** **function** **(**res**){**  console**.**log**(**"Status: "**+**res**.**statusCode**);**  //console.log("Headers: \n", res.headers);  res**.**on**(**'data'**,** **function** **(**data**){**  **var** data3 **=** data**.**toString**();**  //console.log("Response:", data3);  **});**  **});**  req**.**write**(**jsonoutput**);**  req**.**end**();**  req**.**on**(**'error'**,** **function(**e**){**  console**.**log**(**"Error:"**,** e**);**  **});**  **}** |

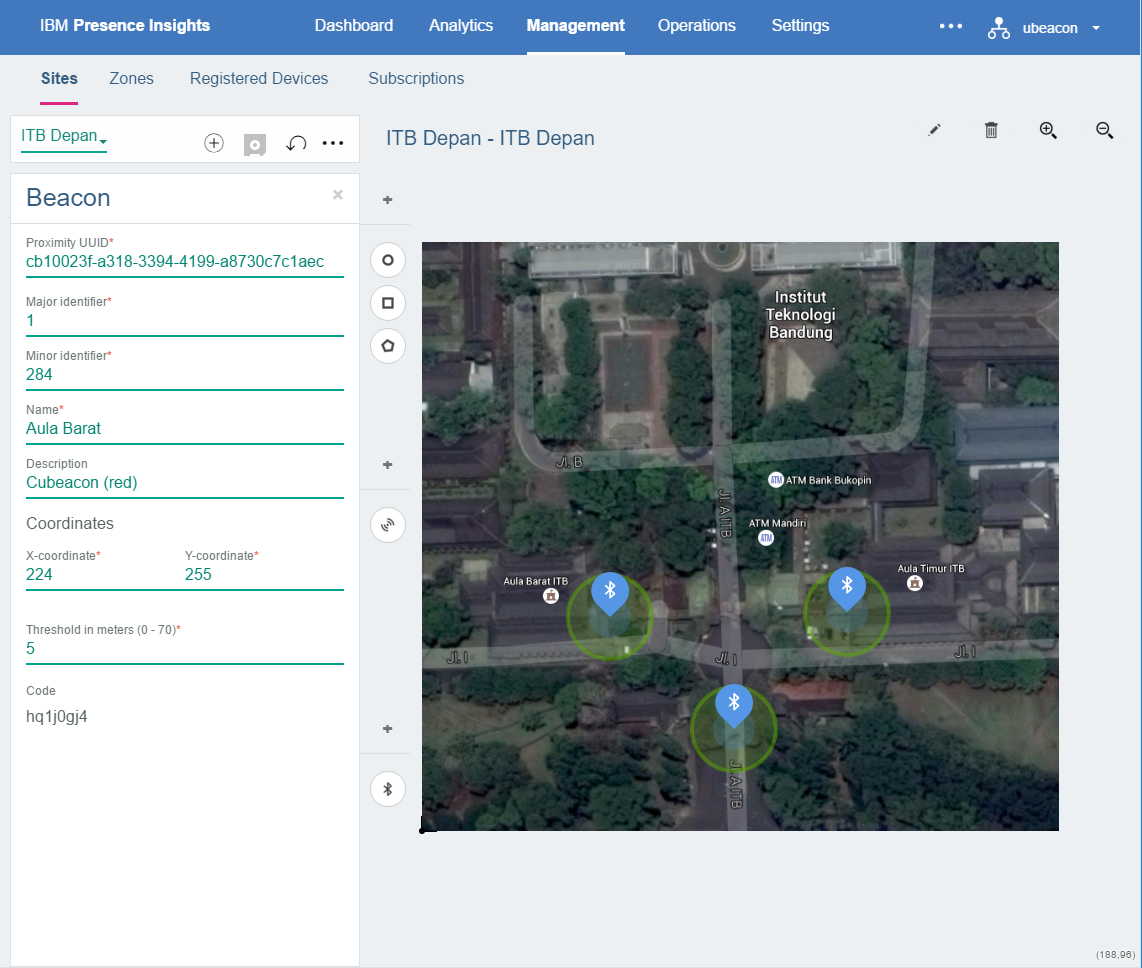
**Tabel 3.10** Detail HTTP POSTS untuk pengiriman data ke PI.

|  |  |
| --- | --- |
| Key | Value |
| *Hostname* | presenceinsights.au-syd.bluemix.net |
| *Port* | 443 |
| *Path* | /conn-beacon/v1/tenants/1g4a05cq/orgs/my1u08tp |
| *Method* | POST |
| *Header* | Content-Type: application/json  Authorization: Basic NTJnMDBrbTpiNXhQdzZKLW5USnE= |
| *Body* | **{**  "bnm"**:** **[**  **{**  "descriptor"**:** "aa:bb:cc:dd:ee:ff"**,**  "detectedTime"**:** "2016-02-02T11:00:00.000Z"**,**  "data"**:** **{**  "proximityUUID"**:** "cb10023f-a318-3394-4199-a8730c7c1aec"**,**  "major"**:** "284"**,**  "minor"**:** "1"**,**  "rssi"**:** 69**,**  "accuracy"**:** 75**,**  "proximity"**:** "Immediate"  **}**  **}**  **]**  **}** |

Respons yang diberikan oleh PI ke *back-end application* berupa *status code*. Bila respons yang diterima berupa *status code* 202, artinya data *beacon* telah berhasil diterima PI. Apabila respons yang diterima berupa *status code* 400, artinya ada kesalahan pada format JSON yang dikirimkan.

Sebelumnya pada PI, konfigurasi *beacon* yang digunakan telah dimasukkan. Konfigurasi pada PI meliputi peta tempat *beacon* akan ditempatkan, titik-titik tempat *beacon* diletakkan, dan data *beacon* untuk setiap titik. Data penempatan *beacon* terpisah berdasarkan *sites*, *floors*,dan *zones*.

Pada gambar berikut ini telah dibuat sebuah konfigurasi dari tiga *beacon* yang akan digunakan dalam implementasi. *Beacon* ini tersebar pada satu *site*, yang berisi satu *floor*, dan tiga *zones*.



**Gambar 3.8** Tampilan IBM Presence Insights ketika mengkonfigurasi beacon pada menu management. © Dokumentasi Penulis.

**Tabel 3.11** *Sites* dan *Zones* yang telah dikonfigurasikan pada PI.

|  |  |  |  |
| --- | --- | --- | --- |
| Sites (code) | Floor (code) | Zones (code) | Beacon (code) |
| ITB Depan  (j51s0wdc) | Ground Floor  (uh1m05g0) | Gerbang Depan  (uq1y08sd)  Radius 10 m | Blue Beacon  (lt1n0597)  Threshold 5 m |
| Aula Bara  (vf1m076u)  Radius 10 m | Red Beacon  (hq1j0gj4)  Threshold 5 m |
| Aula Timur  (vz1r018b)  Radius 10 m | Green Beacon  (ks1v08ia)  Threshold 5 m |

Konfigurasi ini adalah contoh yang digunakan pada saat implementasi. Konfigurasi tidak hanya terbatas pada tiga *beacon* ini. Sebagai contoh, pengembangan yang lebih luas pasti membutuhkan lebih banyak *sites* dengan setiap *sites* memuat beberapa *floor*, setiap *floor* terdapat beberapa *zones*, dan setiap *zones* memuat beberapa *beacon*.

Pada saat data *beacon* masuk, PI secara otomatis akan menghasilkan sebuah paket data JSON lain yang disebut sebagai *event*. JSON ini berisi data lengkap tentang keberadaan perangkat terhadap *beacon*. Paket data ini berisi *sites, floors,* dan *zones* tempat perangkat mendeteksi *beacon*. Selain itu, aplikasi Android dirancang mengirimkan data tentang *beacon* yang terdeteksi setiap 5 menit sehingga PI dapat mengetahui durasi perangkat tersebut berada di sekitar *beacon*. Berikut ini adalah contoh *event* yang dihasilkan oleh PI.

|  |
| --- |
| **{**  "device"**:** **{**  // The following is for a registered device.  // An unregistered device would only contain the following: "descriptor" and "registered"  "data"**:** **{},** // Object  "unencryptedData"**:** **{},** // Object  "name"**:** "VisitorsPhone"**,** // String  "descriptor"**:** "425f34421e7000400f7f93937a694200f2d715d4"**,** // String  "registered"**:** true**,** // Boolean  "blacklist"**:** false // Boolean  **},**  "tenant\_code"**:** "7lf0lv6"**,** // String  "tenantDoc"**:** **{**  "name"**:** "7lf0lv6"**,** // String  "dwellTime"**:** 20000**,** // Number  "deviceTimeout"**:** 10000 // Number  **},**  "org"**:** **{**  "name"**:** "Org"**,** // String  "registrationTypes"**:** **[**  "Internal"**,**  "External"  **],** // Array of unique Strings  "description"**:** null // String or null  **},**  "site"**:** **{**  "name"**:** "Site"**,** // String  "tags"**:** **[],** // Array of unique Strings, may be empty  "address"**:** **{**  "country"**:** ""**,**  "state"**:** ""**,**  "city"**:** ""**,**  "street"**:** ""**,**  "zip"**:** ""  **},** // Object  "description"**:** null**,** // String or null  "timeZone"**:** "America/New\_York" // String  **},**  "floor"**:** **{** // Object, floor follows GeoJson format  "type"**:** "Feature"**,** // String  "geometry"**:** **{**  "type"**:** "Point"**,**  "coordinates"**:** **[**  0**,**  0  **]**  **},** // Object  "properties"**:** **{**  "name"**:** "Lobby"**,**  "z"**:** 1  **},** // Object  "name"**:** "Lobby" // String  **},**  "zone"**:** **{},** // Object, zone follows GeoJson format , may be empty  "zone\_code"**:** "pl36w01mv"**,** // String  "zoneTags"**:** **[],** // Array of unique Strings, may be empty  "site\_code"**:** "b43bm0bmd"**,** // String  "siteTags"**:** **[],** // Array of unique Strings, may be empty  "org\_code"**:** "wi2qo0119"**,** // String  "floor\_code"**:** "qc35g01ch"**,** // String  "detected\_timestamp"**:** 1441717101111**,** // Number  "device\_descriptor"**:** "425f34421e7000400f7f93937a694200f2d715d4"**,** // String  "device\_created"**:** 1445903471963**,** // Number  "registered"**:** true**,** // Boolean  "registrationType"**:** "anonymous"**,** // String  "x"**:** 91**,** // Number  "y"**:** 104**,** // Number  "request\_id"**:** "proximity\_7lf0lv6\_1b6b2f575af441b9b0519f768a05e752"**,** // String  "sensor\_type"**:** "proximity"**,** // String  "state"**:** 1**,** // Number  "dwellPeriod"**:** 0**,** // Number  "dwellPeriodDelta"**:** 0**,** // Number  "dwellId"**:** "aa3dbc20-8f01-11e5-8c99-910b7d395e83"**,** // String  "@docType"**:** "RealTimeEvent"**,** // String  "activity"**:** "enter"**,** // String: "enter","exit",or "dwell"  "subscription\_code"**:** "p5me00vm" // String  **}** |

Pada JSON *event* tersebut, terdapat atribut berupa *floor\_code*, *site\_code*, dan *floor\_code* yang menginformasikan tempat perangkat mendeteksi *beacon* tersebut. Atribut *detected\_timestamp* merupakan waktu ketika perangkat mendeteksi *beacon*. Atribut *state* merupakan status yang menandakan keadaan perangkat tersebut. *State* tersebut dapat berupa *enter*, *dwell*, *exit*, atau *timeout*. Atribut *dwellPeriod* merupakan durasi waktu perangkat tersebut berada di sekitar *beacon*.

Tabel berikut ini akan menjelaskan secara detail tentang atribut *state*.

**Tabel 3.12** *State* dan penjelasannya dari detail *event* yang dikirimkan PI.

|  |  |  |
| --- | --- | --- |
| State | Number | Description |
| *Enter* | 1 | *State* akan bernilai 1 ketika perangkat berada di sekitar *beacon* untuk pertama kalinya |
| *Dwell* | 0 | *State* akan bernilai 0 ketika perangkat sudah berada di sekitar *beacon* selama 5 menit berikutnya |
| *Exit* | –1 | *State* akan bernilai -1 ketika perangkat berada di sekitar yang *beacon* lain sehingga perangkat dianggap sudah tidak berada di sekitar *beacon* terakhir |
| *Timeout* | –2 | *State* akan bernilai -2 ketika perangkat tidak lagi berada di sekitar *beacon* terakhir dan juga tidak berada di sekitar *beacon* lain. |

Pada PI, telah disediakan sebuah antarmuka untuk melihat semua data beserta analisisnya tentang perangkat pengguna yang mendeteksi *beacon*. Data-data ini dapat digunakan untuk kebutuhan analisis kepadatan pengunjung pada titik-titik tertentu di kampus. Berikut ini adalah data yang disajikan pada *dashboard* PI.

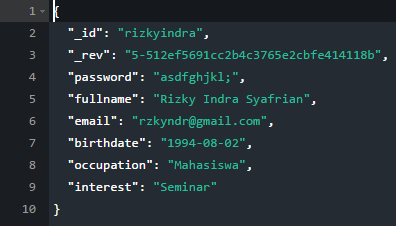
**Tabel 3.13** Data dan analitik yang disediakan PI

|  |  |  |  |
| --- | --- | --- | --- |
| Analytic | Details | | Dashboard |
| Visits | Total Visits | Site Visitors |  |
| Visit by Zone |
| Visit Density Across Time | |  |
| Visit Density on Floor Layout | |  |
| Duration | Average Visit Duration | |  |
| Duration Density Across Time | |  |
| Duration Density on Floor Layout | |  |

# PENGUJIAN

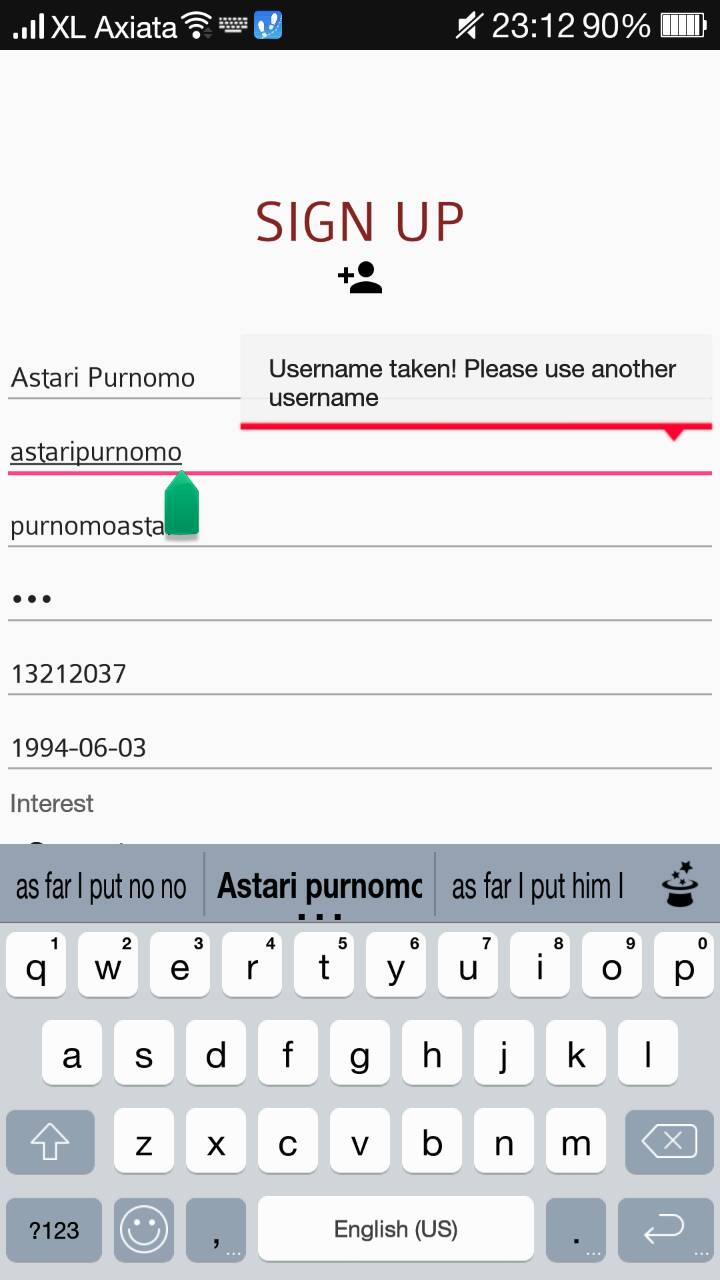
* 1. **Pengujian Sistem Registrasi**

Ketika dilakukan *sign-up* pada aplikasi Android, terlihat dari status balasan bahwa *back-end application* telah menerima data *sign-up.* Selanjutnya adalah penambahan data tersebut ke dalam *database*. Terlihat pada *database* data *sign-up* telah masuk dan menjadi bagian dari *database*.



**Gambar 4.1** Informasi pengguna yang tersimpan pada database. © Dokumentasi Penulis.

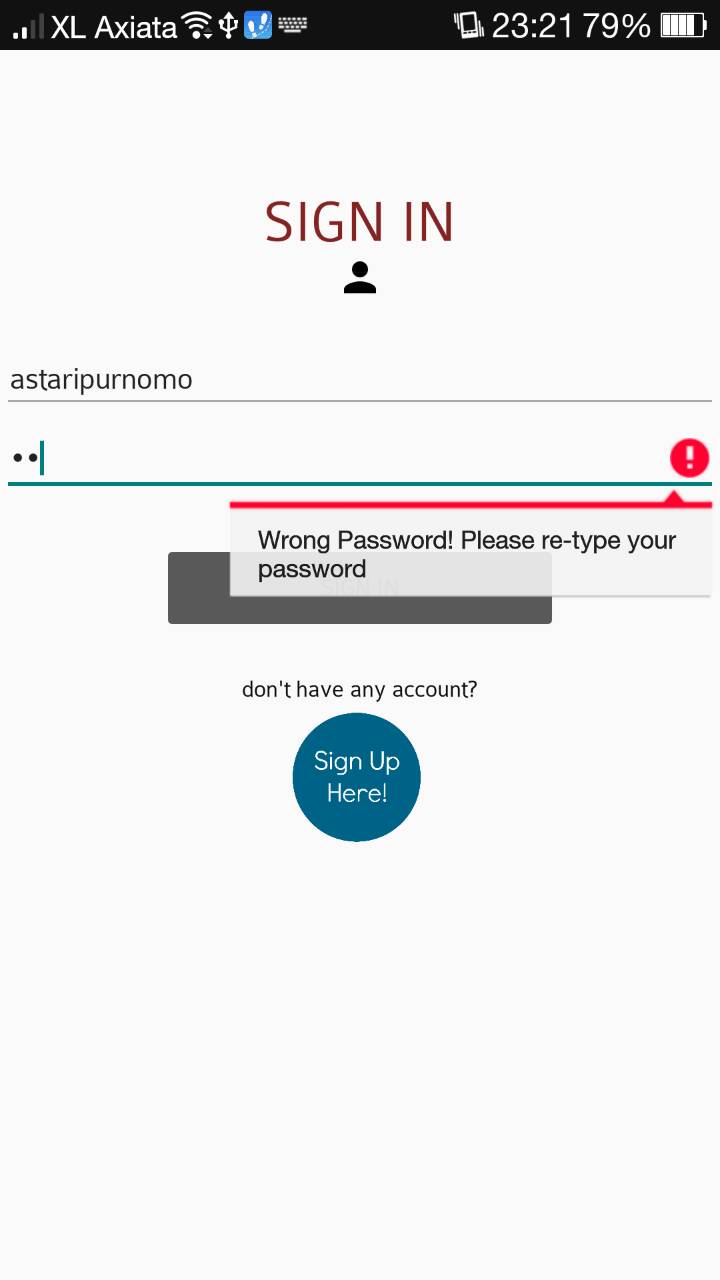
Apabila dilakukan *sign-up* kembali dengan data registrasi yang sama maka *back-end application* akan menolak. Dapat dilihat pada status balasan yang diberikan *back-end application*.



**Gambar 4.2** Tampilan aplikasi menolak registrasi pengguna menggunakan username yang sama. © Dokumentasi Penulis.

Ketika dilakukan *sign-in* pada aplikasi Android, terlihat dari status balasan bahwa *back-end application* telah menerima data *sign-in*. Selanjutnya adalah mencocokkan data *sign-in* dengan *database*. Status balasan akan menunjukkan keberhasilan *sign-in*.

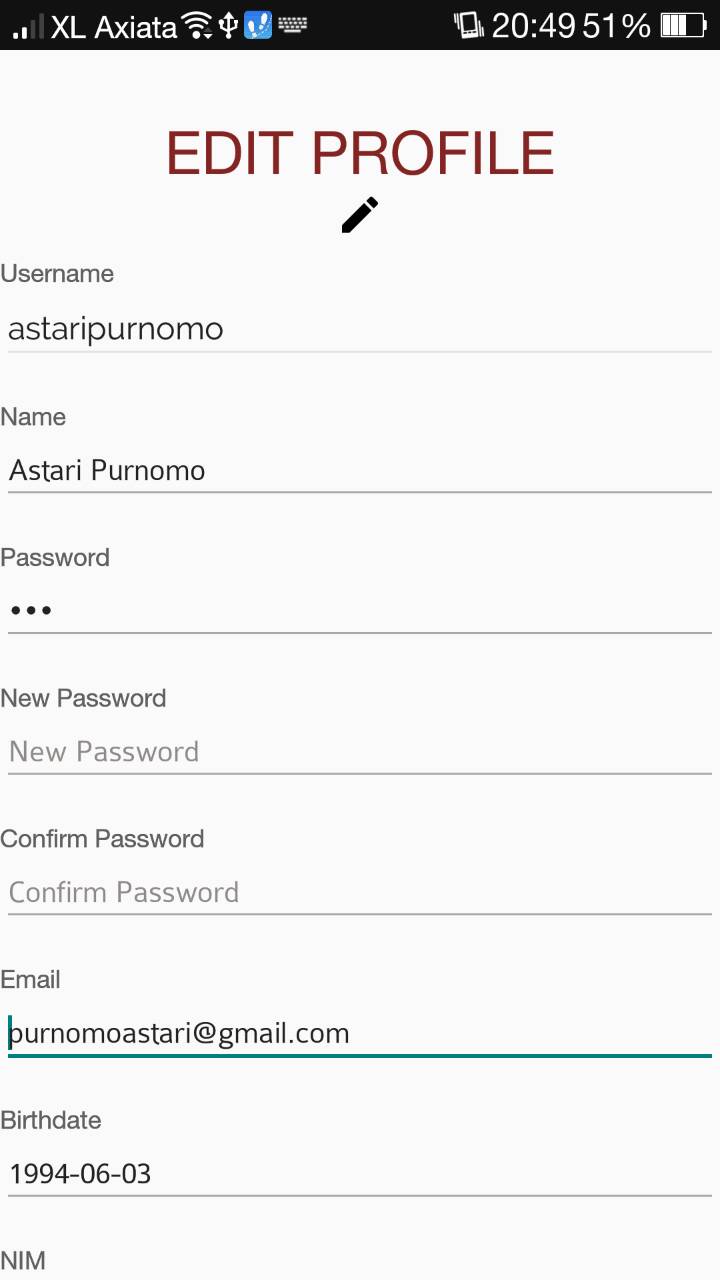
Ketika *sign-in* dilakukan dengan *password* yang tidak sesuai dengan *database*, maka *back-end application* akan menolak proses *sign-in*. Jika *sign-in* dilakukan dengan *username* yang tidak terdaftar, maka *back-end application* akan memberikan status bahwa *username* tidak terdaftar.



**Gambar 4.3** Tampilan aplikasi menolak sign-in karena password tidak cocok dengan database. © Dokumentasi Penulis.

Ketika dilakukan *edit profile* pada aplikasi Android, *back-end application* akan menyesuaikan *password* dan *username* dengan data yang ada pada *database*. Apabila sesuai, maka status balasan bahwa *edit profile* berhasil akan dikirim.

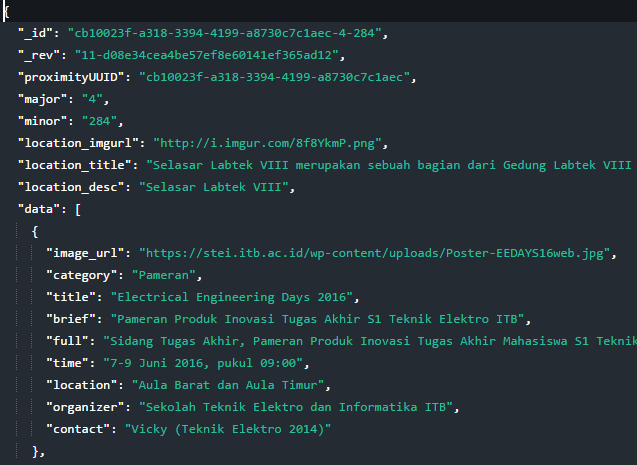
Jika pengguna tidak berkehendak untuk mengganti *password* ketika *edit profile* digunakan, maka pengguna dapat mengosongkan bagian *new password* pada aplikasi Android. *Password* lama akan tetap dapat digunakan dan data lain yang diganti akan berubah pada *database*.



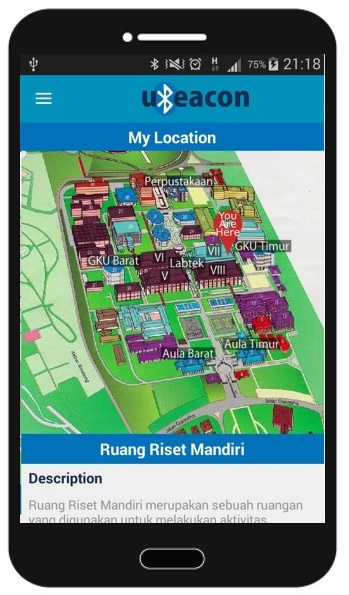
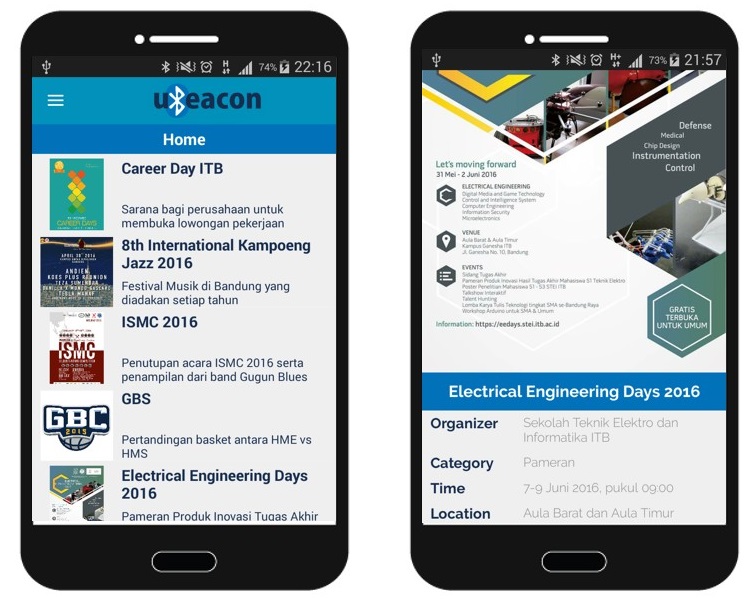
**Gambar 4.4** Tampilan edit profile untuk mengubah data registrasi pengguna. © Dokumentasi Penulis.

* 1. **Pengujian *Beacon Information Request***

Pada saat *mobile application* mendeteksi *beacon* dan melakukan *request* informasi *beacon*, *back-end application* akan menjalankan fungsi *Beacon Information Request*. *Back-end application* akan membalas *request* tersebut dengan informasi yang ada pada *database*. Pada *mobile application* akan ditampilkan informasi *beacon* yang sesuai dengan informasi *beacon* yang tersimpan pada *database*.



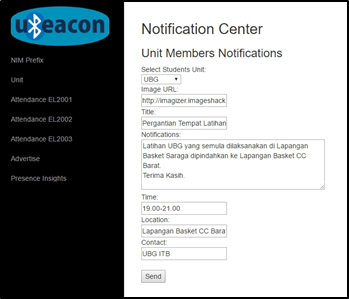
**Gambar 4.5** Informasi beacon yang tersimpan pada database. © Dokumentasi Penulis.

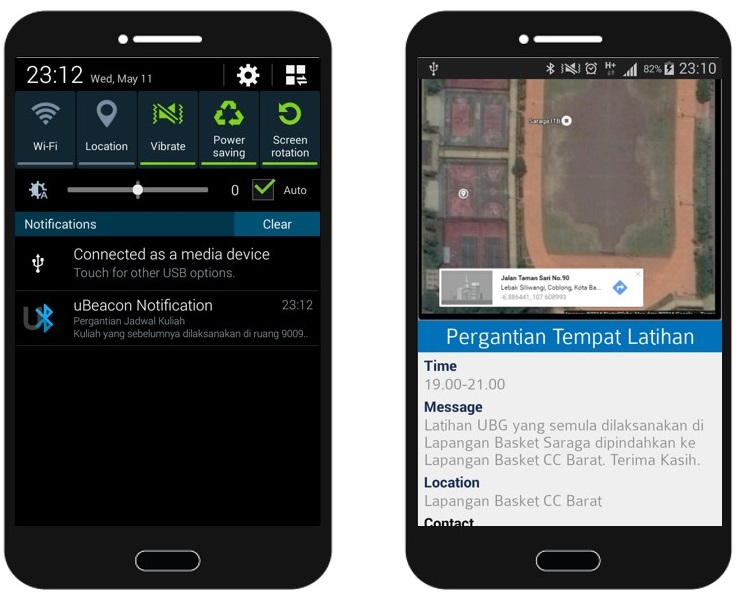
**Gambar 4.6** Aplikasi menampilkan informasi yang sama seperti pada database sesuai dengan beacon yang terdeteksi. © Dokumentasi Penulis.

* 1. **Pengujian *Notification System***

Pada saat notifikasi dikirimkan dari *front-end application* oleh administrator, notifikasi segera diterima oleh pengguna aplikasi yang menjadi penerimanya. Pada contoh di bawah, yang dilakukan adalah pengiriman notifikasi unit mahasiswa. Penerima notifikasi adalah pengguna anggota unit tersebut.



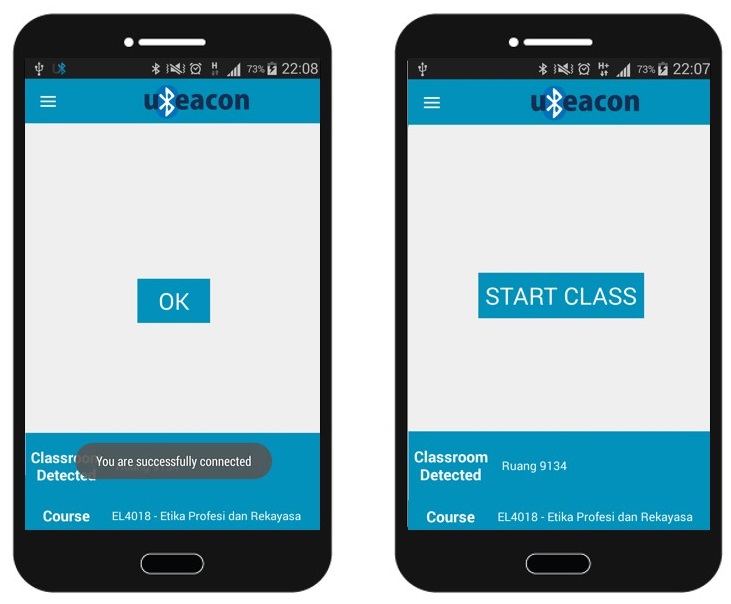
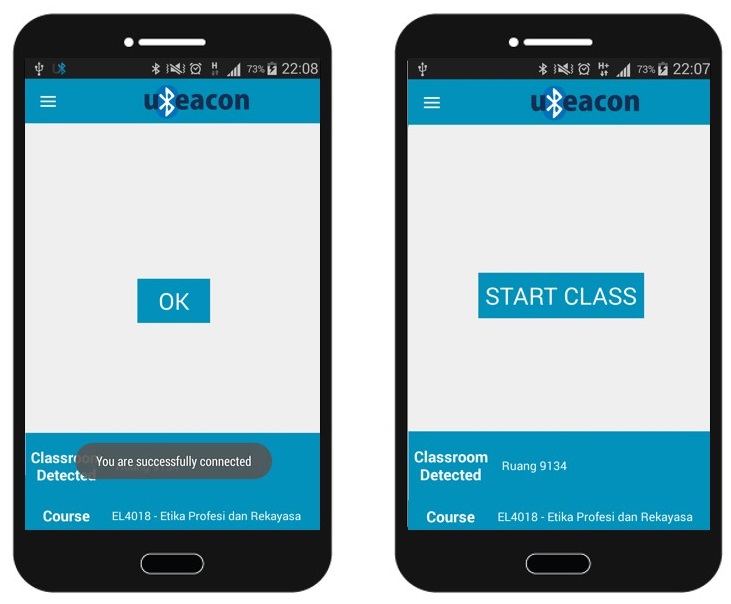
**Gambar 4.7** Front-end application siap mengirimkan notifikasi ke anggota unit mahasiswa. © Dokumentasi Penulis.



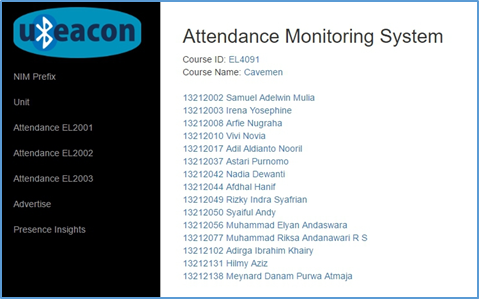
**Gambar 4.8** Notifikasi dengan konten yang sama diterima pengguna. © Dokumentasi Penulis.

* 1. **Pengujian** ***Attendance Monitoring System***

Pada saat *mobile application* mendeteksi *beacon* yang ada di kelas dan pengguna mengaktifkan mode *start class* untuk memulai pengiriman *payload* setiap 5 menit, *front-end application* akan menampilkan data mahasiswa tersebut pada daftar kehadiran kelas.



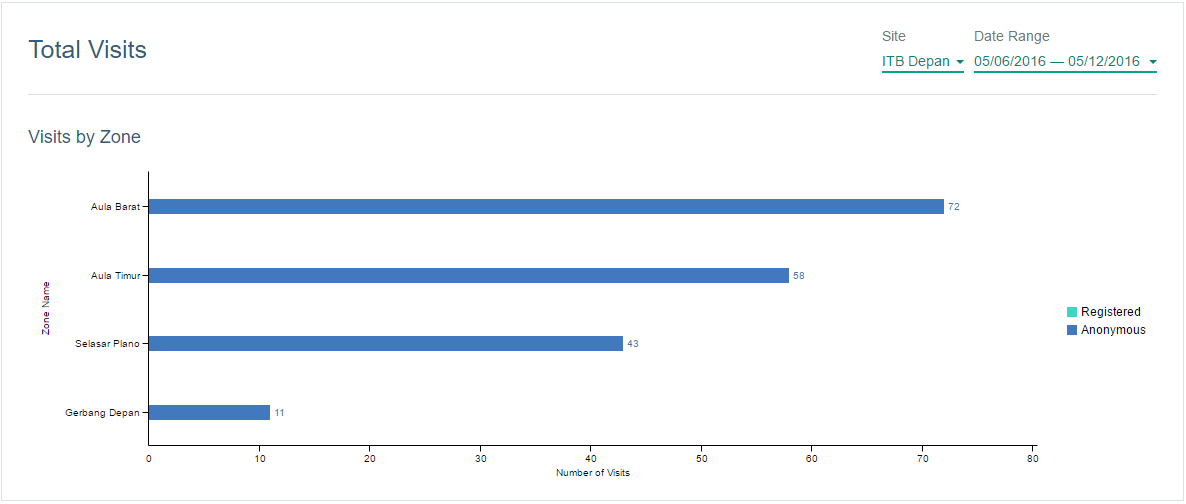
**Gambar 4.9** Mode start class pada mobile application diaktifkan pengguna. © Dokumentasi Penulis.



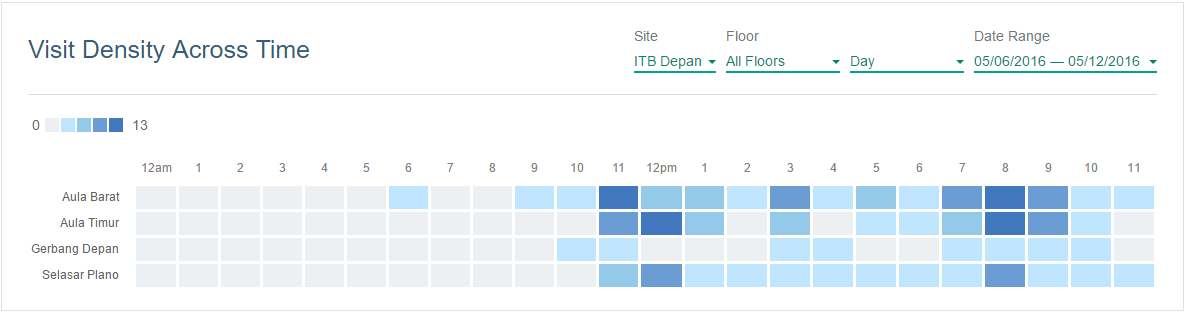
**Gambar 4.10** Daftar nama pengguna yang hadir di kelas muncul pada front-end application. © Dokumentasi Penulis.

* 1. **Pengujian IBM Presence Insights**

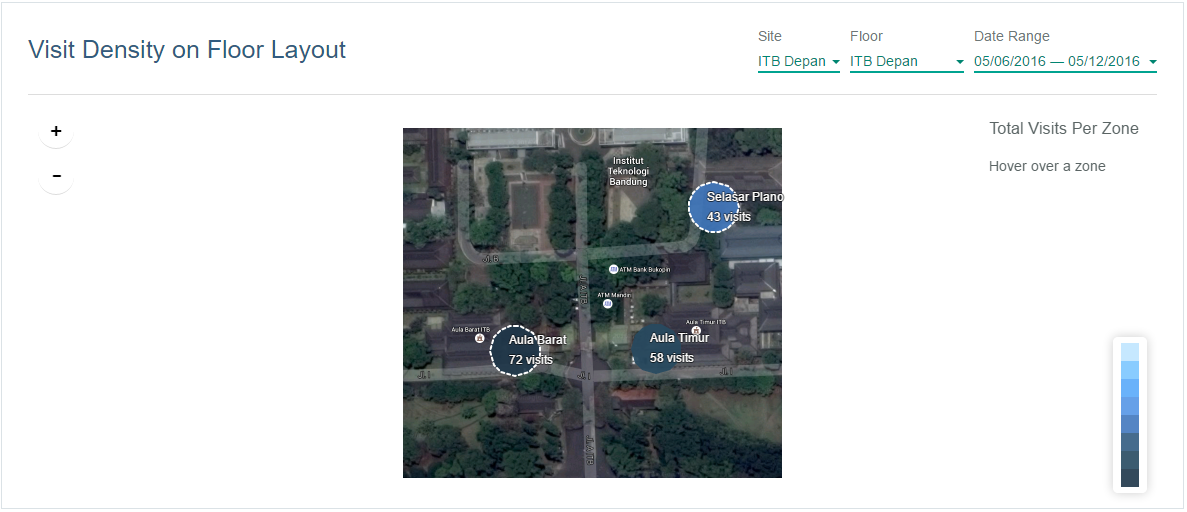
Pada saat *mobile application* mendeteksi *beacon* seperti pada pengujian *Beacon Information Request*, PI akan menerima terusan *payload* tersebut. Laporan dan analisis dari banyak *payload* yang masuk akan ditampilkan pada *dashboard* PI.



**Gambar 4.11** Tampilan grafik jumlah kunjungan pada spot kampus yang telah dipasangi beacon dalam kurun waktu satu minggu. © Dokumentasi Penulis.



**Gambar 4.12** Tampilan persebaran jumlah kunjungan pada jam-jam tertentu dalam kurun waktu satu minggu. © Dokumentasi Penulis.



**Gambar 4.13** Tampilan jumlah kunjungan pada peta lokasi tempat beacon dipasang. © Dokumentasi Penulis.

# KESIMPULAN DAN SARAN

* 1. **Kesimpulan**

Berdasarkan pengujian dan analisis yang dilakukan, dapat disimpulkan:

* + - 1. *Front-end* dan *back-end application* yang dibangun dapat menjadi server bagi aplikasi *smartphone* yang telah dibuat.
      2. Sistem registrasi yang dibuat dapat menyimpan data pengguna aplikasi.
      3. *Back-end* dapat mengembalikan informasi *beacon* sesuai dengan informasi yang ada di *database*.
      4. *Front-end* dapat mengirimkan notifikasi pada pengguna aplikasi spesifik.
      5. *Back-end* dapat memroses data pengguna aplikasi menjadi data kehadiran kelas kuliah.
      6. *Front-end* dapat menampilkan data pengguna aplikasi yang hadir di kelas kuliah.
      7. IBM Presence Insights dapat menampilkan hasil laporan dan analisis keramaian berdasarkan data yang dikirimkan pengguna aplikasi di kampus.
  1. **Saran**

Untuk pengembangan lebih lanjut, sistem yang melibatkan *beacon*, *mobile application*, dan *cloud* PaaS pada lingkungan kampus dapat digunakan

* + - 1. untuk memonitor keberadaan mahasiswa di sekeliling kampus;
      2. memberikan informasi lokasi ruangan kelas;
      3. menjadi *reminder* jadwal perkuliahan;
      4. menjadi *tour guide* untuk mahasiswa baru atau pengunjung kampus;
      5. pengguna tidak terbatas pada mahasiswa tetapi juga untuk dosen dan umum;
      6. transmisi data dari dan ke server juga perlu dilengkapi dengan enkripsi agar keamanan data pengguna terjaga.

Untuk pengembangan sistem di luar lingkungan kampus fungsi yang ditawarkan dapat lebih beragam.

# DAFTAR PUSTAKA

**Ecma International.** Introducing JSON. [Online] [Dikutip: 13 Mei 2016.] http://www.json.org/.

*Flexible Technologies for Smart Campus.* **Van Merode, Dirk, et al. 2016.** Madrid : s.n., 2016. 13th International Conference on Remote Engineering and Virtual Instrumentation (REV). hal. 64-68.

**Google Inc.** Android 6.0 Changes | Android Developers. [Online] [Dikutip: 13 Mei 2016.] http://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-apache-http-client.

**—. 2016.** Mobile App Backend Services - Solutions. *Google Cloud Platform.* [Online] 27 April 2016. [Dikutip: 13 Mei 2016.] https://cloud.google.com/solutions/mobile/mobile-app-backend-services.

**International Business Machines Corporation.** Bluetooth Beacons - Presence Insights Documentation. [Online] [Dikutip: 13 Mei 2016.] https://presenceinsights.au-syd.bluemix.net/pidocs/docs/en/technology/setup\_beacons/#beacon-payload-description.

**—.** Presence Insights Documentation. [Online] [Dikutip: 13 Mei 2016.] https://presenceinsights.ng.bluemix.net/pidocs/docs/en/.

**—. 2015.** What is IBM Bluemix? [Online] 27 April 2015. [Dikutip: 13 Mei 2016.] https://www.ibm.com/developerworks/cloud/library/cl-bluemixfoundry/.

**Jose, Advent. 2015.** 2015, Pengguna Smartphone di Indonesia Capai 55 Juta. [Online] 20 September 2015. [Dikutip: 19 Mei 2016.] http://techno.okezone.com/read/2015/09/19/57/1217340/2015-pengguna-smartphone-di-indonesia-capai-55-juta.

*Managing Large Scale, Ultra-Dense Beacon Deployments in Smart Campuses.* **Wang, Michael dan Brassil, Jack. 2015.** Hong Kong : s.n., 2015. The First International Workshop on Smart Cities and Urban Informatics. hal. 606-611.

**Mourning, Jillian.** How To Use Beacon Technology To Improve The Campus Wi-Fi Experience. [Online] [Dikutip: 19 Mei 2016.] http://www.securedgenetworks.com/blog/using-wireless-beacon-technology-to-improve-the-campus-experience.

**MQTT Community.** FAQ - Frequently Asked Question | MQTT. [Online] [Dikutip: 13 Mei 2016.] http://mqtt.org/faq.

**Perez, Pamela.** Going Mobile: How Student Demand is Changing Campus Wi-Fi Networks. [Online] [Dikutip: 13 Mei 2016.] http://www.securedgenetworks.com/blog/going-mobile-how-student-demand-is-changing-campus-wi-fi-networks.

**PT Media Nusantara Citra Tbk. 2015.** 2015, Pengguna Smartphone di Indonesia Capai 55 Juta : Okezone Techno. [Online] 20 September 2015. [Dikutip: 2016 Mei 2016.] http://techno.okezone.com/read/2015/09/19/57/1217340/2015-pengguna-smartphone-di-indonesia-capai-55-juta.

**R. Fielding, et al. 1999.** RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1. [Online] Juni 1999. [Dikutip: 13 Mei 2016.] https://tools.ietf.org/html/rfc2616.

**Tutorials Point (I) Pvt. Ltd.** Node.js Introduction. [Online] [Dikutip: 13 Mei 2016.] http://www.tutorialspoint.com/nodejs/nodejs\_introduction.htm.

**—.** Node.js Tutorial. [Online] [Dikutip: 13 Mei 2016.] http://www.tutorialspoint.com/nodejs/index.htm.