

**SKRIPSI**  
**IMPELENTASI GRAPHQL PADA SISTEM PENERIMAAN**  
**MAHASISWA BARU UNIVERSITAS DIAN NUSWANTORO**  
**BERBASIS WEB SERVICE**  
***IMPLEMENTATION OF GRAPHQL ON NEW STUDENT***  
***ADMISSION DIAN NUSWANTORO UNIVERSITY BASE ON***  
***WEB SERVICE***

Diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik  
Informatika



Disusun Oleh:

Nama : Arnaz Adiputra  
NIM : A11.2014.08602  
Program Studi : Teknik Informatika-S1

**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS DIAN NUSWANTORO**  
**SEMARANG**  
**2018**

## **PERSETUJUAN SKRIPSI**

Nama : Arnaz Adiputra

NIM : A11.2014.08602

Program Studi : Teknik Informatika

Fakultas : Ilmu Komputer

Judul Tugas Akhir : Implementasi GraphQL pada Sistem Penerimaan mahasiswa Baru Universitas Dian Nuswantoro berbasis Web Service

Tugas Akhir ini telah diperiksa dan disetujui,

Semarang, 27 Juli 2018

Menyetujui:

Pembimbing

Mengetahui:

Dekan Fakultas Ilmu Komputer

**Fahri Firdausillah S.Kom, M.CS**

**Dr. Abdul Syukur**

## **PENGESAHAN DEWAN PENGUJI**

Nama : Arnaz Adiputra

NIM : A11.2014.08602

Program Studi : Teknik Informatika

Fakultas : Ilmu Komputer

Judul Tugas Akhir : Implementasi GraphQL pada Sistem Penerimaan mahasiswa Baru Universitas Dian Nuswantoro berbasis Web Service

Tugas akhir ini telah diujikan dan dipertahankan dihadapan Dewan Penguji pada Sidang tugas akhir tanggal 27 Juli 2018 Menurut pandangan kami, tugas akhir ini memadai dari segi kualitas maupun kuantitas untuk tujuan penganugrahan gelar Sarjana Komputer (S.Kom.)

Semarang, 27 Juli 2018

Dewan Penguji:

**Nova Rijati, SSi, M.Kom**

Anggota 1

**Junta Zeniarja, M.Kom**

Anggota 2

**Dr. Heribertus Himawan, M.Kom**

Ketua penguji

## **PERNYATAAN KEASLIAN TUGAS AKHIR**

Sebagai mahasiswa Universitas Dian Nuswantoro, yang bertanda tangan di bawah ini, saya:

Nama : Arnaz Adiputra

NIM : A11.2014.08602

Menyatakan bahwa karya ilmiah saya yang berjudul :

### **Implementasi *GraphQL* pada Sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro Berbasis *Web Service***

Menyatakan dengan sesungguhnya bahwa karya tulis tugas akhir ini benar-benar saya kerjakan sendiri. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, yang disertai dengan bukti-bukti yang cukup, maka saya bersedia untuk dibatalkan gelar saya beserta hak dan kewajiban yang melekat pada gelar tersebut. Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang

Pada tanggal : 27 Juli 2018

Yang Menyatakan

**(Arnaz Adiputra)**

## **PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai mahasiswa Universitas Dian Nuswantoro, yang bertanda tangan di bawah ini, saya :

Nama : Arnaz Adiputra

NIM : A11.2014.08602

Demi mengembangkan Ilmu Pengetahuan, menyetujui untuk memberikan kepada Universitas Dian Nuswantoro Hak Bebas Royalti Non-Eksklusif (Non exclusive Royalty-Free Right) atas karya ilmiah saya yang berjudul :

### **Implementasi *GraphQL* pada Sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro Berbasis *Web Service***

Beserta perangkat yang diperlukan. Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Dian Nuswantoro berhak untuk menyimpan, mengcopy ulang (memperbanyak), menggunakan, mengelolanya dalam bentuk pangkalan data (database), mendistribusikannya dan menampilkan/mempublikasikannya diinternet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta. Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Dian Nuswantoro, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini. Demikian surat pernyataan ini saya buat dengan sebenarnya.

Semarang, 27 Juli 2018

**(Arnaz Adiputra)**

## UCAPAN TERIMA KASIH

Dengan memanjatkan puji syukur kehadiran Allah SWT. Tuhan yang Maha Pengasih dan Maha Penyayang yang telah melimpahkan segala rahmat, hidayah serta inayah-Nya kepada penulis sehingga laporan tugas akhir dengan judul: “IMPELENTASI GRAPHQL PADA SISTEM PENERIMAAN MAHASISWA BARU UNIVERSITAS DIAN NUSWANTORO BERBASIS WEB SERVICE” dapat penulis selesaikan dengan rencana dukungan dari berbagai pihak yang tidak ternilai besarnya. Oleh karena itu penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, adik-adik dan keluarga yang telah memberikan motivasi penulis dalam pembuatan laporan Tugas Akhir ini.
2. Bapak Prof. Dr. Ir Edi Noersasongko, M.Kom, selaku Rektor Universitas Dian Nuswantoro Semarang.
3. Bapak Dr. Drs. Abdul Syukur, MM, selaku Dekan Fasilkom Universitas Dian Nuswantoro.
4. Bapak Heru Agus Santoso, Ph.D, selaku Ka.Progdi Teknik informatika –S1.
5. Bapak Fahri Firdausillah S.KOM, M.CS selaku pembimbing tugas akhir yang sangat baik, sabar dalam membimbing penulis saat penulis mengalami banyak kesulitan dan sekaligus ketua PSI Udinus atas ijin dan bantuannya selama penelitian yang dilakukan penulis.
6. Teman-teman DOSCOM, MADHANG, yang telah banyak memberikan doa, semangat, dan bantuan kepada penulis.
7. Kepada semua pihak yang namanya tidak dapat di sebutkan satu persatu.

Semoga Tuhan Yang Maha Esa memberikan balasan yang lebih besar kepada beliau-beliau, dan pada akhirnya berharap bahwa penulisan laporan Tugas Akhir ini dapat bermanfaat dan berguna sebagaimana fungsinya.

Semarang, 27 Juli 2018

Penulis

## ABSTRAK

Penerimaan mahasiswa baru Universitas Dian Nuswantoro merupakan langkah awal untuk para calon mahasiswa dapat menuntut ilmu di Universitas Dian Nuswantoro. Jalur reguler dengan mendaftar di web merupakan salah satu cara untuk calon mahasiswa dapat melakukan pendaftaran. Dengan bertambah pesatnya para pendaftar via web dan juga semakin berkembangnya dunia teknologi maka sekretariat penerimaan mahasiswa baru Universitas Dian Nuswantoro juga harus ikut berbenah untuk membangun sebuah sistem penerimaan mahasiswa baru yang lebih ringan dan efisien. Penerapan GraphQL web service pada sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro diharapkan dapat membantu sekretariat penerimaan mahasiswa baru Universitas Dian Nuswantoro dan memenuhi kebutuhan calon mahasiswa yang akan mendaftar. Dengan menggunakan GraphQL *web service, server-side* dapat memberikan data persis seperti *client-side* minta dan dalam proses perhitungan statistik calon mahasiswa menjadi lebih ringan. Hasil dari penelitian ini menunjukkan pengurangan atribut saat melakukan request ukuran yang bandwidth digunakan berkurang hingga 82,32%. Jadi dapat ditarik kesimpulan bahwa proses request dan response menjadi lebih cepat dan ringan karena dari sisi client-side dapat memilih sendiri atribut yang dibutuhkan.

**Kata kunci:** PMB Universitas Dian Nuswantoro, GraphQL web service, *client-side, server-side*



## ABSTRACT

Dian Nuswantoro University new students enrollment is the first step for prospective students to study at Dian Nuswantoro University. One of the ways to register in regular path is filling the online registration form on the admission web of Dian Nuswantoro University. Because of high increament of registrants through the web and the development of technology, the admissions secretariat of Dian Nuswantoro University new student should build more efficient admission system. By using existing technology, Dian Nuswantoro University new admissions system can be more efficient. Implementation of GraphQL web services on new student admissions system Dian Nuswantoro University can support the admission department of Dian Nuswantoro University and comply the needs for student who register. By using GraphQL web services in the server-side, it can provides precise data such as client-side requests and the prospective student's statistics calculation process become easier. The results of this study show a reduction in attributes when requesting a size for which bandwidth is used is reduced to 82.32. so it can be concluded that the plaroses request and response becomes faster and lighter because from the client-side side can choose their own attributes needed.

**Keywords:** PMB Universitas Dian Nuswantoro, GraphQL web service, *client-side, server-side*

## DAFTAR ISI

|  |      |
|--|------|
| PERSETUJUAN SKRIPSI.....   | ii   |
| PENGESAHAN DEWAN PENGUJI.....  | iii  |
| PERNYATAAN KEASLIAN TUGAS AKHIR.....   | iv   |
| PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK<br>KEPENTINGAN AKADEMIS..... | v    |
| UCAPAN TERIMA KASIH.....   | vi   |
| ABSTRAK.....   | viii |
| ABSTRACT.....  | ix   |
| DAFTAR ISI.....  | x    |
| DAFTAR TABEL.....  | xii  |
| DAFTAR GAMBAR.....   | xiii |
| BAB I PENDAHULUAN.....   | 1    |
| 1.1 Latar Belakang.....  | 1    |
| 1.2 Rumusan Masalah.....   | 4    |
| 1.3 Batasan masalah.....   | 4    |
| 1.4 Tujuan Penelitian.....   | 4    |
| 1.5 Manfaat Penelitian.....  | 5    |
| BAB II TINJAUAN PUSTAKA DAN LANDASAN TEORI.....                                  | 6    |
| 2.1 Tinjauan Studi.....  | 6    |
| 2.2 Tinjauan Pustaka.....  | 11   |
| 2.3 Kerangka Pemikiran.....  | 21   |
| BAB III METODE PENELITIAN.....   | 24   |
| 3.1 Instrumen Penelitian.....  | 24   |
| 3.2 Jenis dan sumber Data.....   | 25   |
| 3.3 Teknik Pengumpulan data.....   | 25   |
| 3.4 Metode pengembangan sistem.....  | 26   |
| 3.5 Metode evaluasi.....   | 33   |

|   |    |
|---|----|
| BAB IV RANCANGAN SISTEM DAN IMPLEMENTASI..... | 34 |
| 4.1Pengantar.....                             | 34 |
| 4.2Studi kasus.....                           | 34 |
| 4.3Rancangan sistem.....                      | 37 |
| 4.4Implementasi Sistem.....                   | 49 |
| BAB V HASIL PENELITIAN DAN PEMBAHASAN.....    | 59 |
| 5.1Hasil penelitian.....                      | 59 |
| 5.2Pengujian.....                             | 76 |
| BAB VI KESIMPULAN DAN SARAN.....              | 88 |
| 6.1Kesimpulan.....                            | 88 |
| 6.2Saran.....                                 | 88 |
| DAFTAR PUSTAKA.....                           | 90 |

## DAFTAR TABEL

|   |    |
|---|----|
| Tabel 2.1 Tinjauan Pustaka.....   | 8  |
| Tabel 2.2 Kerangka Pemikiran.....   | 21 |
| Tabel 4.1 Tabel matriculants.....   | 40 |
| Tabel 4.2 Tabel registrationGroups.....   | 41 |
| Tabel 4.3 Tabel lastEducations.....   | 42 |
| Tabel 4.4 Tabel origins.....  | 42 |
| Tabel 4.5 Tabel matriculantPrograms.....  | 43 |
| Tabel 4.6 Tabel programs.....   | 43 |
| Tabel 4.7 Tabel faculties.....  | 44 |
| Tabel 4.8 Tabel users.....  | 44 |
| Tabel 5.1: Selishi latensi dari kedua request dengan atribut yang berbedas..... | 87 |

## DAFTAR GAMBAR

|  |    |
|--|----|
| Gambar 2.1 Client-Server Model.....  | 12 |
| Gambar 2.2 Contoh syntax HTML.....   | 13 |
| Gambar 2.3 Mendeskripsikan skema pada GraphQL dengan type.....                                     | 16 |
| Gambar 2.4 Mendeskripsikan relasi type pada GraphQL.....   | 17 |
| Gambar 2.5 Query graph.....  | 17 |
| Gambar 2.6 Contoh syntax JSON.....   | 19 |
| Gambar 3.1 Kerangka sistem penerimaan mahasiswa baru Universitas Dian<br>Nuswantoro saat ini.....  | 29 |
| Gambar 3.2 Kerangka Backend API Penerimaan Mahasiswa Baru Universitas<br>Dian Nuswantoro.....      | 31 |
| Gambar 4.1 Activity Diagram Pendaftaran.....   | 37 |
| Gambar 4.2 Activity Diagram ganti status calon mahasiswa.....                                      | 38 |
| Gambar 4.3 ERD basis data sistem penerimaan mahasiswa baru Universitas Dian<br>Nuswantoro.....     | 39 |
| Gambar 4.4: Query SQL daftar matriculant baru.....   | 46 |
| Gambar 4.5: Query SQL melihat data calon mahasiswa.....  | 46 |
| Gambar 4.6: Query SQL mengganti status calon mahasiswa.....  | 47 |
| Gambar 4.7: Query SQL menampilkan data pendaftar berdasarkan jurusan.....                          | 47 |
| Gambar 4.8: Query SQL menampilkan daftar calon mahasiswa yang mundur<br>berdasarkan jurusan.....   | 48 |
| Gambar 4.9: Query SQL menampilkan daftar calon mahasiswa berdasar jurusan<br>dan sekolah asal..... | 48 |
| Gambar 4.10: Skema basis data sistem PMB Universitas Dian Nuswantoro.....                          | 50 |
| Gambar 4.11 Mutation GraphQL menyimpan data user baru.....   | 51 |
| Gambar 4.12 Query GraphQL menampilkan data user.....   | 52 |
| Gambar 4.13 Mutation GraphQL mengganti status user.....  | 53 |
| Gambar 4.14 Query GraphQL pencarian seluruh calon mahasiswa.....                                   | 54 |

|  |    |
|--|----|
| Gambar 4.15 Query GraphQL jumlah calon mahasiswa berdasarkan tahun dan bulan.....                      | 55 |
| Gambar 4.16 Query GraphQL menampilkan jumlah calon mahasiswa berdasarkan tipe pendaftaran.....         | 56 |
| Gambar 4.17 Hasil GraphQL menampilkan daftar calon mahasiswa berdasarkan status dan jurusan.....       | 57 |
| Gambar 4.18 Hasil GraphQL menampilkan daftar calon mahasiswa berdasarkan asal sekolah dan jurusan..... | 58 |
| Gambar 5.1 Developer tool graphql.....   | 59 |
| Gambar 5.2 Request mutation createMatriculant.....   | 60 |
| Gambar 5.3 Response mutation createMatriculant.....  | 60 |
| Gambar 5.4 Request fungsi findMatrulant.....   | 61 |
| Gambar 5.5 Response fungsi findMatrulant.....  | 62 |
| Gambar 5.6 Reqeust mutation changeStatusMatriculant.....   | 63 |
| Gambar 5.7 Responsemutation changeStatusMatriculant.....   | 63 |
| Gambar 5.8 Request query statMatriculant.....  | 64 |
| Gambar 5.9 Response query statMatriculant.....   | 64 |
| Gambar 5.10 Request query matriculantPerMonth.....   | 65 |
| Gambar 5.11 Responsequery matriculantPerMonth.....   | 66 |
| Gambar 5.12 Request query sortMatriculant.....   | 67 |
| Gambar 5.13 Response query sortMatriculant.....  | 67 |
| Gambar 5.14 Request dari fungsi matriculantByProgram.....  | 68 |
| Gambar 5.15 Response dari fungsi matriculantByProgram.....   | 68 |
| Gambar 5.16 Request fungsi query matriculantByLastEdu.....   | 69 |
| Gambar 5.17 Response fungsi query matriculantByLastEdu.....  | 70 |
| Gambar 5.18 Form pendataran mahasiswa baru Universitas Dian Nuswantoro. .                              | 71 |
| Gambar 5.19 Halaman admin.....   | 72 |
| Gambar 5.20 Statistik jumlah pendaftar berdasarkan tipe pendaftaran.....                               | 72 |
| Gambar 5.21 Statistik jumlah pendaftar berdasarkan bulan dan tahun sekarang..                          | 73 |
| Gambar 5.22 List data pendaftar.....   | 74 |

|  |    |
|--|----|
| Gambar 5.23 Detail data calon mahasiswa.....                           | 75 |
| Gambar 5.24 Potongan kode query statMatriculant.....                   | 77 |
| Gambar 5.25 Flow graph fungsi statMatriculant.....                     | 78 |
| Gambar 5.26 Potongan kode fungsi filterMonth.....                      | 79 |
| Gambar 5.27 Flow graph fungsi filterMonth.....                         | 80 |
| Gambar 5.28 Request fungsi query matriculantAll.....                   | 84 |
| Gambar 5.29 Jumlah bandwidth dan ukuran respon dengan 21 atribut.....  | 85 |
| Gambar 5.30 Jumlah bandwidth dan ukuran pada query matriculantAll..... | 86 |

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Penerimaan mahasiswa baru Universitas Dian Nuswantoro merupakan gerbang awal bagi calon mahasiswa untuk dapat berkuliah di Universitas Dian Nuswantoro. Universitas Dian Nuswantoro membuka tiga jenis jalur pendaftaran salah satunya ialah jalur reguler. Jalur Reguler adalah jalur penerimaan mahasiswa baru untuk jenjang strata satu (S-1), Program Vokasi (D3), dan Diploma 4 melalui tes potensi akademik yang dilaksanakan secara mandiri atau kolektif. Pendaftaran Jalur Reguler dapat dilakukan secara langsung di sekretariat Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro atau online[1].

Jalur reguler memiliki dua jenis prosedur, salah satunya pendaftaran online berbasis web. Untuk menggunakan pendaftaran online berbasis web calon mahasiswa diwajibkan untuk membaca tata tertib pendaftaran online yang sudah disediakan, mengunjungi website resmi Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro, mengisi formulir pendaftaran, mentransfer biaya pendaftaran di rekening yang sudah ditentukan, mengunggah bukti pembayaran, menunggu jadwal ujian, melakukan ujian dan keluar hasil ujian.

Jalur Reguler dengan prosedur pendaftaran online berbasis web merupakan bentuk perkembangan teknologi yang dimiliki Sekretariat Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro. Perkembangan teknologi di dunia juga diimbangi peningkatan pengguna internet di tahunnya. Peningkatan tersebut dibuktikan oleh “The Statistics Portal”, di tahun terakhir kenaikan mencapai sembilan belas juta pengguna internet[2].



Data yang dipublikasi oleh “We Are Social” dan “Hootsuite” pada tahun 2017 yang penulis kutip dari Kadata, Indonesia merupakan negara terbesar nomer satu didunia dalam hal pertumbuhan penggunaan internet sebesar 51% dalam kurun waktu satu tahun[3]. Hal ini juga diimbangi dengan peningkatan pencarian di google dengan perangkat *mobile*, dari tahun 2013 sampai tahun 2014 kenaikan mencapai 1,8%, kenaikan ini akan bertambah tiap tahunnya[4].

Dengan meningkatnya penggunaan perangkat mobile untuk akses internet yang begitu pesat, maka hal tersebut menjadi penting untuk Sekretariat Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro untuk mulai mengembangkan aplikasi berbasis *mobile apps* agar dapat memudahkan calon mahasiswa untuk melakukan pendaftaran dengan jalur online dengan menggunakan perangkat *mobile*.

Pembangunan *mobile apps* memiliki beberapa komponen utama, salah satunya pembuatan *Restful Web Service*. *Restful Web Service* merupakan sebuah *service* yang digunakan untuk menghubungkan aplikasi klien dengan server[5]. Untuk sekarang *restful* merupakan solusi terbaik untuk pembangunan sebuah API yang nantinya akan menjadi jembatan bagi aplikasi klien dengan server, *syntax*-nya yang simple, *request* dan *response* yang mudah dibuat dan di parsing membuat *restful* menjadi populer dan digunakan dibanyak project[6]. Terlepas dari kelebihanannya *resful* juga memiliki beberapa kekurangan.

Kekurang *restful* adalah dalam project sekala besar memerlukan banyak *endpoint* dalam satu halaman. Hal ini ditinjau dari rata-rata untuk melakukan *HTTP request* membutuhkan hampir 1 detik, sedangkan satu halam dari aplikasi klien membutuhkan lebih dari satu *endpoint*[7]. Menggunakan *resful* juga rentan terhadap *vulnerability*, karena *endpoint*

yang dimiliki *restful* berupa url, dimana dengan mengetahui url dari setiap *endpoint* hal tersebut akan menjadi rentan. Disisi yang sama, kecepatan akses internet dengan perangkat mobile lebih lambat dari WiFi karena *latency* tertinggi yang dimiliki 3G mencapai 3500ms dan 4G mencapai 600ms[8] . Hal lain yang menjadi penting khususnya pengguna aplikasi mobile ialah kecepatan akses internet di Indonesia masih jauh tertinggal dari negara lain. Dari survei yang dilakukan Akamai Technologies pada kuartal IV 2016, dari 15 negara Indonesia berada peringkat 12 dengan rata rata kecepatan 6.7Mbps[9]. Dari data-data diatas terbukti *restful* kurang bagus untuk diimplementasikan di dalam sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro.

Dari sekian kekurangan yang dimiliki *Restful* penulis menawarkan teknologi baru yaitu *GraphQL*. *GraphQL* merupakan sebuah *service* yang digunakan untuk menghubungkan aplikasi klien ke server dengan konsep baru[10], dimana pengembang aplikasi klien dapat meminta *response* dari *service GraphQL* tanpa atribut yang tidak dibutuhkan dalam *request*-nya. Dengan *GraphQL* pula pengembang bisa menghemat endpoint yang dibutuhkan. Teknologi ini dikembangkan oleh Facebook pada tahun 2015 dan sudah digunakan oleh beberapa perusahaan besar seperti Facebook, Github, Pinterest dan masih banyak lagi.

Dengan masalah diatas dan fakta yang penulis paparkan, maka penulis tertarik untuk melakukan penelitian mengenai “Implementasi GraphQL pada Sistem Penerimaan mahasiswa Baru Universitas Dian Nuswantoro berbasis Web Service”. Untuk membangun *Backend API* untuk sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro yang memiliki sedikit endpoint . Sehingga diharapkan dapat membantu kinerja dari Sekretariat Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro.

## 1.2 Rumusan Masalah

Berdasarkan apa yang sudah penulis jabarkan dilatar belakang, maka rumusan dalam masalah ini adalah bagaimana membangun Backend API untuk aplikasi mobile Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro dengan minimum *endpoint*.

## 1.3 Batasan Masalah

Agar penulisan penelitian ini sesuai dengan tujuan awal, maka perlu diberikannya batasan masalah sebagai berikut:

1. Penulis menggunakan *GraphQL* yang digunakan sebagai metode *Web Service* dengan hasil berupa *JSON*.
2. Aplikasi yang penulis kembangkan dalam penelitian ini berbasis *Backend API* untuk aplikasi mobile.
3. *Backend API* meliputi pendaftaran, pendaftaran ulang, dan laporan statistik pada sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro.

## 1.4 Tujuan Penelitian

Dari rumusan masalah yang di jelaskan sebelumnya, tujuan dari kegiatan penelitian ini adalah membangun *Backend API* untuk aplikasi mobile Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro dimana hemat penggunaan endpoint sehingga bisa mengurangi *latency* pada saat aplikasi mobile melakukan *request* ke *Backend API* yang penulis buat.

## **1.5 Manfaat Penelitian**

### **1.5.1 Bagi Penulis**

- a. Penulis dapat mengimplentasikan pengetahuan yang didapat selama masa kuliah.
- b. Menambah wawasan penulis terkait implementasi GraphQL yang penulis dapatkan semasa kuliah.
- c. Memenuhi salah satu persyaratan kelulusan starta satu (S1), Program Studi Teknik Informatika, Fakutlas Ilmu Komputer, Universitas Dian Nuswantoro.

### **1.5.2 Bagi Universitas Dian Nuswantoro**

- a. Menadapatkan alternatif solusi dalam sistem penerimaan mahasiswa baru berbasis *Web Service*.
- b. Dapat dijadikan tambahan informasi dan rekomendasi topik penelitian kepada mahasiswa dengan minta yang sama.

## **BAB II**

### **TINJAUAN PUSTAKA DAN LANDASAN TEORI**

#### **2.1 Tinjauan Studi**

Pada tahun 2016 Facebook mengumumkan sebuah teknologi untuk melakukan request data ke Web API dengan konsep berbeda dibanding REST dengan nama GraphQL. Penelitian tentang “An Initial Analysis of Facebook’s GraphQL Language” yang diangkat oleh Olaf Hartig dan Jorge Pérez, merupakan sebuah penelitian awal dari GraphQL[10]. Konsep dari GraphQL sendiri ialah GraphQL tidak mendefinisikan model data secara langsung, melainkan secara implisit menganggap data model yang diimplementasikan berupa tampilan berbasis graph dari beberapa database yang mendasarinya. Kueri yang digunakan GraphQL menyerupai bentuk dari JSON(Javascript Object Notation). Penelitian ini menghasilkan sebuah teknologi baru untuk mengakses data pada Web API dengan konsep Graph.

Kit Gustavsson dan Erik Stenlund pada penelitan “Efficient Data Communication Between a Web Client and a Cloud Environment” menjelaskan perbedaan arsitektur antara REST yang selama ini menjadi standar pembangunan sebuah web API dengan graphql yang baru baru ini muncul, penelitian ini tidak membahas mana yang lebih bagus dan mana yang lebih buruk, tetapi pembahasan lebih ke perbedaan teknik dari keduanya[11]. Penelitian ini memiliki dua tujuan utama, yang pertama untuk meneliti dan menunjukan perbedaan antara *REST* dan *GraphQL*. Kedua, berdasarkan dari hasil penelitian, harus terlebih dahulu membuat model keputusan untuk menentukan jenis teknik mana yang akan digunakan dan pengaruh saat menggunakan teknik tersebut dalam pengembangan *Web API*. Efek yang akan didapat ketika pihak pengembang menggunakan GraphQL

ialah, pengembang harus bergantung dengan *dependencies* GraphQL sedangkan *dependencies* tersebut merupakan *external dependencies*. Dan jika pengembang lebih memilih menggunakan REST API, pengembang perlu menentukan terlebih dahulu bahasa yang akan digunakan tetapi tidak perlu bergantung pada *external dependencies*. Dari sisi lain para pengembang juga harus mempertimbangkan segi performa, yang mana dari hasil penelitian ini menunjukkan GraphQL dapat mengurangi beban dari server maupun client.

Penelitian “Implementing GraphQL as a Query Language for Deductive Databases in SWI-Prolog Using DCGs, Quasi Quotations, and Dicts” yang ditulis Mike Bryant membahas tentang apa GraphQL tersebut, perbandingan antara GraphQL dengan REST dan implementasi GraphQL pada SWI-Prolog yang memiliki Deductive Databases[12]. GraphQL merupakan sebuah *application layer* yang digunakan untuk query data dan manipulasi yang dikembangkan oleh Facebook. GraphQL ini juga dapat memproses data dari berbagai sumber database, contohnya menggabungkan data dari database relasional dengan database NoSql. Berbeda dengan REST, GraphQL hanya menyediakan sebuah *endpoint* yang fleksibel dan dapat dilakukan proses query pada *endpoint* tersebut. Penelitian ini telah menghasilkan SWI-Prolog Versi 7 atau yang disebut *GraphQL.pl* yang telah menggunakan server GraphQL.

EHRI atau The European Holocaust Research Infrastructure merupakan sebuah lembaga yang mendukung para peneliti Holocaust dalam bentuk infrastruktur digital. EHRI menyediakan akses data online mengenai informasi Holocaust tersebut melalui portal online, dan semua hal yang dapat membantu para peneliti Holocaust [13]. Penelitian yang diangkat oleh Mike Bryant berjudul “GraphQL for Archival Metadata: An Overview of the EHRI GraphQL API” membahas tentang penerapan GraphQL pada server

EHRI[14]. Database yang digunakan EHRI merupakan Graph database Neo4j-based, hal ini memberikan banyak keuntungan ketika mengimplementasikan GraphQL kedalam Graph database, beberapa keuntungannya ialah karena kedua teknologi tersebut sama sama memiliki konsep graph dan kueri dari GraphQL mewarisi dari kueri Neo4j-based. Tujuan dari penelitian ini ialah untuk mempermudah para peneliti dalam proses ekstrak data untuk tujuan penelitian.

*Tabel 2.1 Tinjauan Pustaka*

| <b>No.</b> | <b>Nama</b>                       | <b>Tahun</b> | <b>Judul</b>   | <b>Metode</b> | <b>Hasil</b>  |
|------------|-----------------------------------|--------------|--|---------------|---|
| 1.         | Olaf<br>Hartig,<br>Jorge<br>Pérez | 2016         | An Initial<br>Analysis of<br>Facebook's<br>GraphQL<br>Language | Graph         | Penelitian ini<br>menghasilkan sebuah<br>teknologi baru untuk<br>mengakses data pada Web<br>API menggunakan kueri<br>berbentuk Graph. |

|    |  |      |  |   |  |
|----|--|------|--|---|--|
| 2. | Kit Gustavsson, Erik Stenlundngompolan , | 2016 | Efficient Data Communication Between a Webclient and a Cloud Environment   | Representational State Transfer( REST), GraphQL | Pada penelitian ini menghasilkan, jika pengembang lebih memilih menggunakan GraphQL, pengembang harus bergantung pada external dependencies dengan pertimbangan performa yang lebih cepat ketimbang REST API. Tetapi jika pengembang lebih memilih menggunakan REST API pengembang bisa tidak bergantung pada external dependencies. |
| 3. | Falco Nogatz, Dietmar Seipel             | 2017 | Implementing GraphQL as a Query Language for Deductive Databases in SWI-Prolog Using DCGs, Quasi Quotations, and Dicts | GraphQL   | Pada penelitian ini menghasilkan SWI-Prolog versi 7 atau disebut GraphQL.pl yang sudah menggunakan atau mengimplementasi GraphQL pada sistem tersebut.   |



|    |             |      |  |         |   |
|----|-------------|------|--|---------|---|
| 4. | Mike Bryant | 2017 | GraphQL for Archival Metadata: An Overview of the EHRI GraphQL API | GraphQL | Pada penelitian ini menghasilkan response data yang lebih mudah diekstrak bagi pada pengguna(dalam hal ini peneliti) dan juga API yang dihasilkan dari implementasi GraphQL menjadi lebih cepat dikarenakan konsep kueri API-nya memiliki konsep yang sama dengan sist database yang dimiliki EHRI. |
|----|-------------|------|--|---------|---|

## 2.2 Tinjauan Pustaka

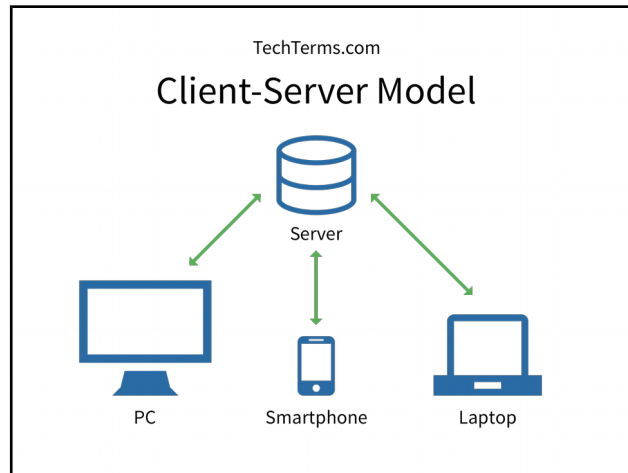
### 2.2.1 Interoperabilitas

Berkembangnya dunia teknologi informasi menyebabkan sebuah sistem informasi bisa dibangun dengan berbagai sumber atau *platform* yang berberda. Dengan adanya hal ini menimbulkan sebuah tuntutan dimana sebuah informasi bisa melakukan integrasi yang baik dengan sistem yang lain. Sedangkan setiap sistem informasi memiliki bahasa pemrograman yang berbeda, database yang berbeda, karakter yang berbeda dengan sistem informasi lainnya.

Adanya masalah diatas menjadi proses pertukaran informasi dari setiap sistem informasi menjadi terganggu, namun sudah bisa diatasi dengan adanya interoperabilitas. Interoperabilitas ialah sebuah standar yang memungkinkan sebuah sistem bisa bertukar informasi dengan sistem lain tanpa ada hambatan dari segi perbedaan *platform*, struktur data, bahasa pemrograman, database[15].

### 2.2.2 Client-Server Model

Merupakan sebuah aplikasi yang terdistribusi dimana memiliki server yang bekerja sebagai penyedia layanan atau *resource* dan Client yang menggunakan layanan tersebut[16]. Secara umum relasi yang dimiliki server dengan *client* ialah one-to-many, jadi sebuah server bisa diakses banyak *client* dalam waktu yang bersamaan. Berikut gambaran dari Client-Server Model. Biasanya *client* disebut juga *frontend* dan server disebut *backend* dimana kedua buah sistem ini saling berhubungan dan menjadi sebuah sistem yang utuh.



Gambar 2.1 Client-Server Model

- a. Client-side: merupakan sebuah aplikasi sisi klien yang dijalankan dengan sebuah *device* yang menerima inputan dari pengguna. Aplikasi sisi klien ini juga menyiapkan data atau informasi yang dibutuhkan pengguna, setelah pengguna memasukan informasi, data akan dikirim ke server atau yang biasanya disebut *request*.
- b. Server-side: merupakan sebuah aplikasi sisi server yang mana berfungsi sebagai menerima *request* dari aplikasi sisi klien yang langsung memproses *request* tersebut dan mengirimkan tanggapan sesuai dengan permintaan aplikasi sisi klien atau biasa disebut *response*.

### 2.2.3 World Wide Web

Perkembangan akses internet sangatlah pesat, hal ini menjadi salah satu bukti bahwa teknologi juga ikut berkembang[2]. Disisi yang sama pertumbuhan sistem informasi juga sangat cepat. Dengan hal ini seluruh sistem informasi lebih gampang untuk diakses. Seluruh sistem informasi yang dapat diakses menggunakan web browser disebut dengan halaman

web(*web page*). Dalam bahasa ilmiah halaman web disebut juga World Wide Web atau biasa disingkat dengan WWW[17].

#### 2.2.4 HTML

Sebuah teknologi informasi berbasis situs web tidak bisa terlepas dari teknologi bernama HTML. HTML merupakan teknologi dasar untuk membangun sebuah halaman web(*web page*). HTML digunakan untuk mendefinisikan atau mentranslasikan konten dari halaman web tersebut, seperti link, paragraf, gambar, heading, dan lain sebagainya[18]. Berikut merupakan contoh syntax dari HTML.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title></title>
5    </head>
6    <body>
7
8    </body>
9  </html>
10

```

Gambar 2.2 Contoh syntax HTML

#### 2.2.5 HyperText Transfer Protocol

Merupakan sebuah protokol *application layer* untuk mengirim atau menerima sebuah dokumen seperti HTML dan lain lain. HTTP digunakan untuk menyambungkan antara web browser dan web server. HTTP juga digunakan sebagai penghubung antara *client-server model*, dimana *client*

meminta tanggapan(*response*) dengan menggunakan permintaan(*request*) [19].

#### 2.2.6 Web Service

Merupakan salah satu bentuk Client-Server model yang termasuk ke dalam Interoperabilitas dengan melakukan komunikasi melalui World Wide Web(WWW) dan HyperText Transfer Protocol (HTTP). Web Service menyediakan sebuah layanan yang dapat diakses oleh semua platform dan kerangka kerja[20]. Web service dapat menerima dan menyimpan informasi dalam format seperti HTTP, XML, SSL, SMTP, SOAP, dan JSON.

#### 2.2.7 REST API

Merupakan sekumpulan fungsi yang mana developer dapat melakukan kegiatan request dan response[21]. Ada enam aturan dimana sebuah sistem dikatakan REST API, berikut aturan aturan tersebut[11].

- c. Client-Server : Secara arsitektur REST memisahkan pemrosesan sistem menjadi dua komponen. Server merupakan komponen yang menyediakan layanan dan menanggapi permintaan untuk service tersebut. Client merupakan komponen yang terhubung ke server untuk melakukan permintaan ke server.
- d. Stateless : Server tidak melihat status sesi dari Client. Setiap Request yang dikirim melalui Client harus berisi seluruh informasi yang dibutuhkan agar server dapat mengerti apa yang harus dikirim ke Client.
- e. Cacheable : Response yang dikirim oleh server harus cacheable. Hal ini bertujuan untuk menghindari request yang tidak diperlukan.

- f. Uniform Interface : Dengan perbedaaan komponen dari sistem REST untuk melakukan komunikasi dari kedua komponen memerlukan standar yang sama(Uniform Interface). Hal ini juga mengurangi efisiensi dalam mengirim informasi, karena informasi yang merupakan bentuk standar sedangkan dari pihak aplikasi client memiliki kebutuhan yang berbeda.
- g. Layered System : Sistem ini berada di layer yang berbeda. Satu layer hanya bisa berinteraksi dengan layer terdekatnya. Tetapi dari komponen komponen sistem tidak perlu mengerti satu sama lain, asalkan keduanya bekerja dengan baik maka komunikasi data juga akan bekerja.

#### 2.2.8 GraphQL

Didalam dunia *web service* sering sekali pengembangan sebuah sistem dibangun menggunakan REST, dimana REST sendiri ialah merupakan sekumpulan fungsi yang dapat melakukan *request* dan *response* ke server. Tetapi REST memiliki beberapa kelemahan, kelemahan-kelemahan tersebut sudah dijelaskan diatas. Tahun 2015 Facebook perusahaan yang bergerak dibidang teknologi meluncurkan sebuah teknologi dengan nama GraphQL.

GraphQL merupakan sebuah bahasa query API untuk mengakses data yang ada. GraphQL menyediakan deskripsi data yang lengkap dan mudah dimengerti oleh API, dapat mengakses data persis seperti apa yang diinginkan pengguna[22]. GraphQL hanya memberikan sebuah alamat yang nantinya aplikasi klien akan terhubung dengan alamat tersebut dan melakukan proses query. Yang perlu dilakukan pertama kali untuk melakukan proses query didalam GraphQL ialah mendeskripsikan skema

graph-nya , berikut merupakan contoh gambar untuk mendeskripsikan skema graph,

```
type User {  
  name: String!  
  username: String!  
}
```

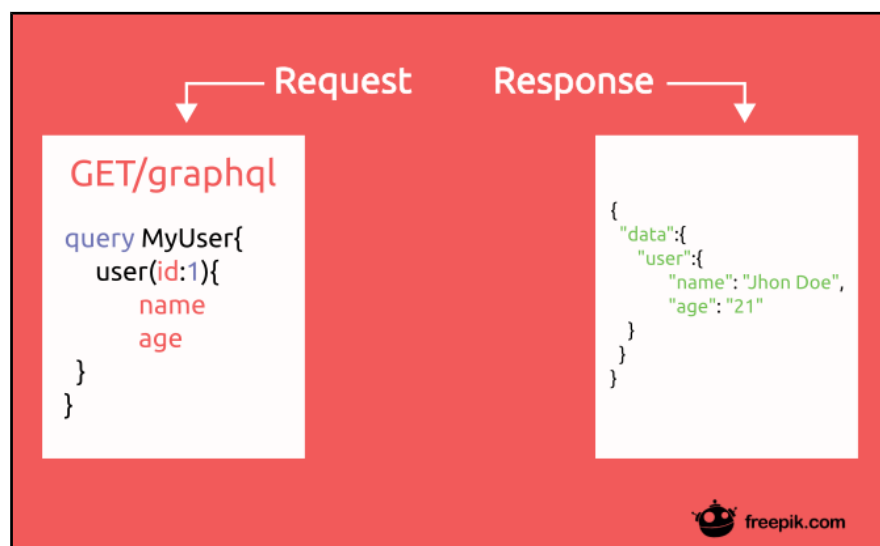
Gambar 2.3 Mendeskripsikan skema  
pada GraphQL dengan type

gambar diatas merupakan skema user yang mana memiliki dua *field* yang mana dari kedua *field* ini bersifat wajib ada atau tidak boleh kosong, *field* yang bersifat wajib ini ditandai dengan tanda ! Pada akhir pendefinisian tipe data *field* tersebut. Didalam GraphQL dapat melakukan relasi antar skema, sebagai contoh skema user diatas memiliki relasi dengan skema *document* dibawah ini

```
type Document {  
  
  title: String!  
  
  content: String!  
  
  author: User!  
  
}
```

Gambar 2.4 Mendeskripsikan relasi type pada GraphQL

gambar diatas merupakan skema document yang berelasi dengan skema user, untuk merelasikan antar skema hanya perlu menuliskan *field* dengan nama skema yang direlasikan sebagai tipe data *field* tersebut. Query yang digunakan graphql sendiri ialah berbentuk graph.



Gambar 2.5 Query graph



Ada tiga jenis *method* dari graphql itu sendiri, ialah:

1. *Query* digunakan untuk meminta data yang dibutuhkan dari server. Tidak seperti REST yang mana struktur respon data yang diminta selalu didefinisikan terlebih dahulu. *Query* graphql memungkinkan sisi klien untuk memutuskan atribut mana yang akan diminta.
2. *Mutation* digunakan untuk membuat data baru, memperbarui data yang sudah ada dan menghapus data yang sudah ada (CUD). Secara dasar mutation sama seperti *Query* tetapi ketika ingin melakukan *CUD* harus diawali dengan kata kunci *Mutation*.
3. *Subscriptions* digunakan untuk membuat data yang diminta memperbarui secara *realtime*. Hal ini memungkinkan sisi klient menerima informasi sebarumungkin.

#### 2.2.9 JSON

Merupakan format pertukaran data yang ringan, mudah untuk dibaca dan di tulis oleh manusia, mudah diurai menjadi bahasa mesin. JSON berasal dari subnet bahasa pemrograman JavaScript Standard ECMA-262 3dr Edition-Desember 1999. JSON merupakan format text yang independen namun sangat familiar bagi kebanyakan bahasa pemrograman lain seperti C, C++, C#, Java, JavaScript, Perl, Python, PHP dan masih banyak lagi. Hal ini menjadikan JSON sebagai bahasa pertukaran data yang ideal[23]. Berikut merupakan contoh dari syntax JSON:

```
{
  "empid": "SJ011MS",
  "personal": {
    "name": "Smith Jones",
    "gender": "Male",
    "age": 28,
    "address": {
      "streetaddress": "7 24th Street",
      "city": "New York",
      "state": "NY",
      "postalcode": "10038"
    }
  },
  "profile": {
    "designation": "Deputy General",
    "department": "Finance"
  }
}
```

www.kodingmadesimple.com

Gambar 2.6 Contoh syntax JSON

#### 2.2.10 Deductive Databases

Merupakan sebuah sistem database yang dapat mengambil sebuah kesimpulan berdasarkan aturan atau fakta yang ada. Deductive database ialah sebuah basis data relasional yang mendukung pemodelan data yang lebih kompleks[24]. Berikut merupakan beberapa perbedaan antara *deductive database* dengan logika pemrograman:

- a. Order sensitivity dan procedurality: pada logika pemrograman cara mengeksekusi kode tergantung pada urutan kode tersebut dan juga bisa tergantung dari rule yang sudah dibuat oleh sebuah bahasa pemrograman tertentu. Hal tersebut bertujuan untuk efisiensi dari sebuah logika pemrograman itu sendiri. Dalam *deductive database* cara mengeksekusi tidak bergantung pada urutan aturan dan fakta.
- b. Function symbols: dalam bahasa pemrograman untuk membangun sebuah fungsi yang kompleks digunakan *function symbol*. Sedangkan di *deductive database* tidak mengenal itu.

#### 2.2.11 Database Relational

Merupakan sebuah sistem database yang mengatur datanya menggunakan tabel yang saling dihubungkan dengan relasi, dalam sebuah tabel terdapat baris dan kolom seperti tabel semestinya. Dari setiap baris data terdapat atribut kunci untuk menjadi pembeda dari data lain. Bentuk database ini bertujuan salah satunya untuk menghindari data yang redundan[25]. Data yang terdapat pada database ini biasa disebut record

### 2.3 Kerangka Pemikiran

Berikut merupakan kerangka pemikiran dari penelitian ini,

*Tabel 2.2 Kerangka Pemikiran*

| Problem   |
|---|
| <ul style="list-style-type: none"> <li>• Belum adanya Backend API dari sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro</li> </ul>  |
| Approach  |
| <ul style="list-style-type: none"> <li>• Pembuatan Backend API menggunakan GraphQL untuk sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro</li> </ul>  |
| Development   |
| <ul style="list-style-type: none"> <li>• Sever Side: Node.js dengan menggunakan framwork express.js</li> </ul>  |
| Implementation  |
| <ul style="list-style-type: none"> <li>• Server PMB Universitas Dian Nuswantoro menyediakan informasi yang diperlukan oleh pengguna aplikasi dan juga berperan sebagai server utama untuk menyimpan data penerimaan mahasiswa baru</li> </ul> |
| Evaluation and Validation   |
| <ul style="list-style-type: none"> <li>• Mengecek seluruh fitur yang ada pada server GraphQL</li> </ul>   |
| Result  |
| <ul style="list-style-type: none"> <li>• Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro berbasis GraphQL Web Service</li> </ul>  |

*a. Problem*

Sebelum melakukan penelitian ini, peneliti sudah melakukan kajian studi terlebih dahulu dengan topik terkait. Dari sinilah penulis menemukan permasalahan untuk melandasi penelitian ini. Permasalahan yang ditemukan ialah, belum adanya sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro berbasis aplikasi mobile.

*b. Approach*

Dari ditemukannya permasalahan diatas, penulis mencoba untuk menemukan solusi untuk memecahkan permasalahan Backend API yang memiliki sedikit endpoint dan mengirimkan response data berupa JSON yang persis seperti aplikasi mobile butuhkan.

*c. Development*

Metode untuk memecahkan masalah sudah dibahas sebelumnya, selanjutnya akan Backend API dari sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro akan diimplementasikan menggunakan bahasa pemrograman Node.js dengan *framework* Express.

*d. Implementation*

Setelah tahap *development* selesai maka menghasilkan sebuah aplikasi berupa Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro yang siap digunakan untuk aplikasi

mobile dan juga data dari penerimaan mahasiswa baru akan disimpan di server Backend API tersebut.

*e. Evaluation and Validation*

Tahap selanjutnya merupakan tahap pengecekan semua *response* yang ada pada Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro.

*f. Result*

Setelah seluruh tahap terselesaikan maka hasil dari penelitian ini berupa Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro yang nantinya diharapkan dapat menyelesaikan masalah yang sudah diurai diatas.

## **BAB III**

### **METODE PENELITIAN**

#### **3.1 Instrumen Penelitian**

Dalam melakukan sebuah penelitian tentu saja diperlukan berbagai macam perangkat yang digunakan, yaitu:

##### **3.1.1 Kebutuhan Software**

Software atau perangkat lunak yang digunakan untuk menyelesaikan penelitian ini ialah:

1. Sistem Operasi yang digunakan adalah Xubuntu 16.04 LST 64 bit.
2. Text editor yang digunakan untuk menulis kode Backend API adalah Atom 1.22 64 Bit.
3. Postgres sebagai penyimpanan informasi yang diperlukan.
4. *Nginx* sebagai mesin untuk menjalankan aplikasi.

##### **3.1.2 Kebutuhan Hardware**

Hardware atau perangkat keras yang dibutuhkan ialah:

1. Komputer yang digunakan Thinkpad T440P.
2. Prosesor Intel Core i5 generasi 4 vPro.
3. Kapasitas RAM 4GB.
4. Penyimpanan SSD Samasung EVO 850 250GB.

### **3.2 Jenis Dan Sumber Data**

Penulis telah mengumpulkan beberapa jenis data sebagai acuan penelitian, data data tersebut dapat dikelompokkan menjadi dua bagian, sebagai berikut:

1. Data Primer

Data primer adalah data yang dijadikan objek penelitian dan diperoleh secara langsung dari sumber penelitian tersebut. Data primer diperoleh dengan cara melakukan observasi secara langsung melalui wawancara, data yang didapatkan menjadi acuan untuk pembangunan sistem.

2. Data Sekunder

Data sekunder adalah data yang dijadikan landasan teori dan penunjang atau pelengkap data primer yang ada. Data sekunder didapatkan dari studi literatur dan dokumen penelitian terkait sebelumnya.

### **3.3 Teknik Pengumpulan Data**

Metode yang digunakan untuk mengumpulkan data dalam melakukan penelitian ini adalah, sebagai berikut:

- 3.3.1 Wawancara

Dengan cara berkomunikasi secara langsung dengan pihak-pihak yang berkaitan dengan pengembangan Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro mengenai alur dari sistem dan segala kebutuhan user, dengan begitu peneliti akan mengetahui secara detail kesulitan dan kebutuhan yang dimiliki pada saat melakukan pengembangan sistem tersebut.



### 3.3.2 Studi Pustaka

Metode ini digunakan dengan cara mencari dan memahami teori-teori yang ada pada literatur terkait dengan penelitian yang dilakukan, salah satunya pada jurnal “Efficient Data Communication Between a Web Client and a Cloud Environment”[11]. Jurnal ini menjelaskan tentang perbedaan REST dengan GraphQL pada penerapan Web Service. Dari segi performa GraphQL lebih unggul dari pada REST karena GraphQL dapat memberikan response JSON ke aplikasi klien dengan atribut yang sama persis seperti apa yang dibutuhkan.

### 3.4 Metode Pengembangan Sistem

Pembangunan sebuah sistem yang baik tidaklah terlepas dari sebuah metode pengembangan sistem yang sesuai dengan kebutuhan sistem tersebut. Dalam menyelesaikan penelitian tentang Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro, penulis menggunakan metode *Extreme Programming* (Pemrograman Ekstrim).

Pemrograman Ekstrim ini merupakan sebuah metode pengembangan sistem yang paling sering digunakan oleh banyak pengembang sistem karena metode ini menyederhanakan beberapa tahapan dalam proses pengembangan tersebut sehingga menjadi lebih fleksibel dan adaptif. Ada lima langkah dalam mengembangkan sistem dengan menggunakan metode ini, berikut adalah macamnya:

#### 3.4.1 Planning / Perencanaan

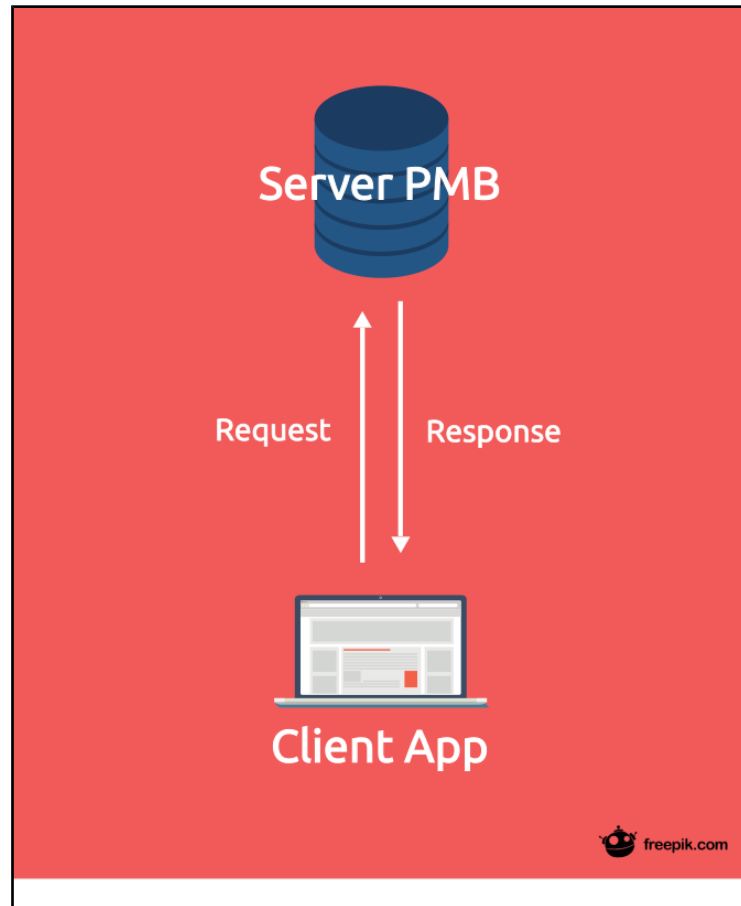
Perencana atau *planning* merupakan tahapan awal dimana peneliti mengumpulkan seluruh kebutuhan, fitur utama, keluaran dari sistem (*output*), dan fungsionalitasnya. Berikut adalah kebutuhannya:

1. Sistem yang dikembangkan merupakan Backend API atau aplikasi sisi server(*server-side*).
2. Fitur utama dari Backend API ini merupakan untuk menyimpan dan memproses data dari aplikasi mobile.
3. Backend API yang penulis kembangkan menggunakan teknologi web service GraphQL.
4. Pengambilan atau pengolahan data dari Backend API menggunakan *query* berbentuk *graph*.
5. Data yang dikeluarkan dari Backend API berformat JSON.

#### 3.4.2 Design / Perancangan

Perancangan dalam pemrograman ekstrim ini memiliki prinsip yaitu penyederhanaan atau *simplicity*. Perancangan yang sederhana selalu memakan waktu yang singkat dibanding perancangan yang kompleks. Jika perancangan sederhana masih menemui kesulitan maka bisa menggunakan solusi *spike* dimana pengembang bisa langsung menerapkan atau mengimplementasikan prototipe perancangan dan dilanjutkan evaluasi. Rancangan sistem penerimaan mahasiswa baru yang lama kurang lebih seperti berikut:





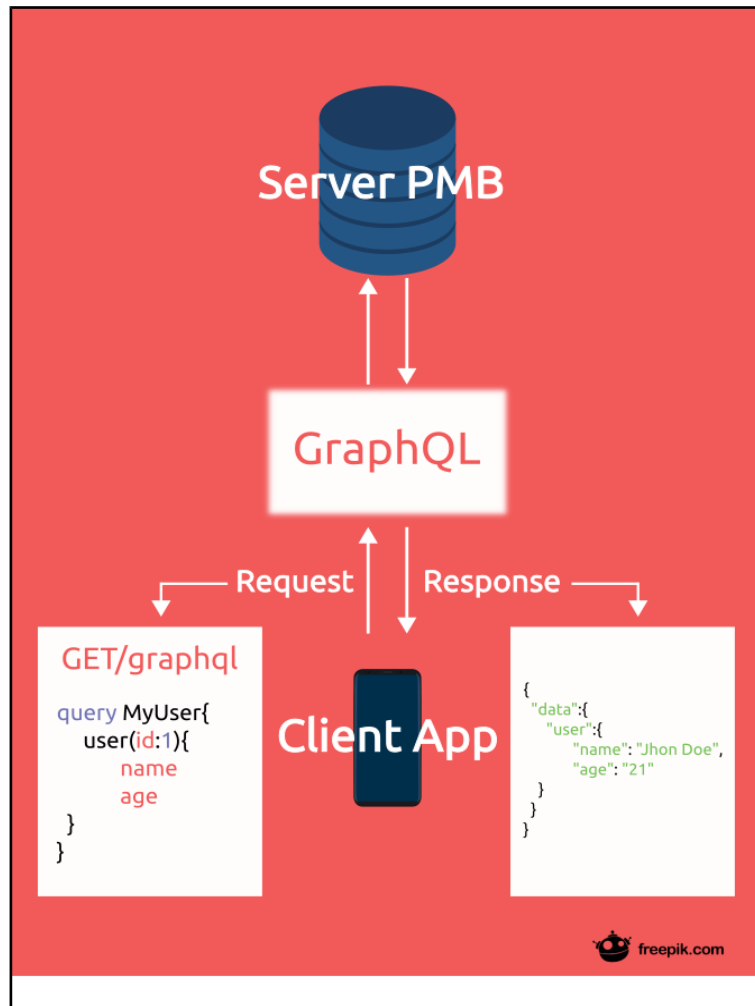
Gambar 3.1 Kerangka sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro saat ini

Alur dari sistem Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro lama dijelaskan sebagai berikut:

1. Server PMB: memiliki tugas untuk menyimpan seluruh data yang digunakan oleh sistem ini. Pada sistem yang lama sistem operasi yang digunakan ialah Centos OS dan databasenya menggunakan Mysql.

2. Request dan Response: agar aplikasi dapat melakukan request dan response dibutuhkan aplikasi sisi server yang ditulis menggunakan PHP versi 5.6
3. Client App: aplikasi sisi klien pada sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro menggunakan *template engine* dari PHP jadi baik dari aplikasi sisi server dan sisi klien berada dalam satu aplikasi.

Perancangan dalam Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro akan dijelaskan pada gambar dibawah:



Gambar 3.2 Kerangka Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro

Alur dari sistem Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro baru dijelaskan sebagai berikut:

1. Server PMB: memiliki tugas untuk menyimpan seluruh data yang digunakan oleh sistem ini. Server menggunakan sistem operasi Ubuntu Server dan database yang digunakan postgresSQL.

2. GraphQL: berfungsi sebagai web service dimana menerima *request* dan *response* dari aplikasi klien. Kode yang digunakan untuk membuat GraphQL menggunakan Node.js versi 8.9.
3. App Client: bertugas untuk melakukan permintaan informasi yang ada pada database dan menerima informasi tersebut. Dalam penelitian ini peneliti tidak membangun aplikasi klien.
4. Request: ini merupakan suatu tindakan dimana aplikasi klien meminta sebuah informasi dari database dengan menggunakan format *query graph*. Setiap request menggunakan GraphQL sisi klien dapat memilih atribut apa saja yang digunakan.
5. Response: merupakan balasan dari tindakan permintaan aplikasi klien ke database berformat JSON, dimana balasan ini bermuat informasi yang sama persis seperti apa yang diminta oleh aplikasi klien. *Response* yang diterima setelah melakukan *request* merupakan atribut yang diminta.

#### 3.4.3 Coding / Pengkodean

Setelah tahap perencanaan dan perancangan, selanjutnya masuk ke tahap pengkodean yang harus sesuai dengan tahap perancangan yang sudah ditulis diatas. Kali ini penulis akan menggunakan bahasa pemrograman Node.js untuk pembangunan Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro. Penulis menggunakan bahasa pemrograman Node.js karena menurut sebuah situs *benchmark* Node.js lebih unggul dibanding dengan bahasa pemrograman PHP[27].

#### 3.4.4 Testing / Pengujian

Selanjutnya merupakan tahap pengujian Backend API, pengujian ini dilakukan dengan cara mengecek seluruh *response* dari Backend API Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro apakah sudah sesuai dengan yang dibutuhkan aplikasi klien dan juga melakukan pengecekan setiap fungsi menggunakan *white-box testing*. White-box testing menggunakan metode kompleksitas siklomatik pada beberapa fungsi yang dianggap penulis fatal. Setelah semua sudah sesuai maka Backend API yang sudah dibangun siap untuk diluncurkan.

### 3.5 Metode Evaluasi

Metode evaluasi yang digunakan oleh penulis kali ini adalah *Black-Box Testing*. *Black-Box Testing* merupakan bentuk pengecekan sistem berdasarkan spesifikasi kebutuhan dari sistem itu sendiri dan tanpa melakukan pengecekan kode. *Black-Box Testing* murni melakukan pengecekan berdasarkan dari tampilan pengguna[28]. Selain menggunakan *black-box testing* penulis juga akan menggunakan *white-box testing* pada penelitian kali ini. *White-box testing* merupakan pengujian berbasis path. Metode ini merupakan salah satu pengujian perangkat lunak untuk menjamin suatu statement dalam setiap jalur independen program minimal dieksekusi sekali. Penulis hanya akan menguji fungsi yang dianggap merupakan fungsi yang fatal atau beresiko tinggi ketika terjadi kesalahan. Pengujian menggunakan *integration testing* dimana sistem yang akan diuji harus sudah selesai dari tahap pembangunan secara utuh. Pengujian dilakukan dengan memasukan inputan ke setiap *endpoint* dan dibandingkan dengan keluaran yang dibutuhkan.



## **BAB IV**

### **RANCANGAN SISTEM DAN IMPLEMENTASI**

#### **4.1 Pengantar**

Pada pembahasan kali ini penulis akan menjelaskan tentang rancangan sistem dan memaparkan hasil implementasi dari metode yang diusulkan untuk menyelesaikan masalah yang telah dibahas. Tujuan dari pembahasan ini ialah untuk membuktikan GraphQL *webservice* dapat menjadikan proses pengiriman data dari aplikasi server ke aplikasi klien lebih efisien.

#### **4.2 Studi Kasus**

*Monolithic Architecture* merupakan sebuah arsitektur aplikasi dimana seluruh kode baik dari pemanggilan basis data, logika, kalkulasi dan juga tampilan aplikasi berada dalam satu proyek. Arsitektur ini masih banyak digunakan karena dengan menggunakan arsitektur monolith ini para pengembang aplikasi lebih mudah untuk membangun sebuah sistem, mudah dalam proses testing (biasanya menggunakan *end-to-end testing*), mudah untuk *deployment*.

Disisi lain *monolithic architecture* memiliki banyak kekurangan. Ukuran dari proyek berarsitektur monolith relatif besar karena seluruh komponen aplikasi berada dalam satu proyek. Besarnya ukuran proyek akan menurunkan kecepatan dalam proses *start-up*. Selain itu dengan seluruh komponen aplikasi berada dalam satu proyek akan mempersulit proses pembuatan fitur baru karena terlalu kompleks. Harus melakukan *redploy* ketika pengembang melakukan update sedangkan untuk melakukan *start-up* aplikasi yang menggunakan arsitektur ini memakan waktu yang cukup lama,

hal ini akan sangat mengganggu ketika load dari aplikasi sedang tinggi tingginya, interoperabilitas dan masih banyak kekurangan dari arsitektur ini.

#### 4.2.1 Sistem Penerimaan Mahasiswa Baru Universitas Dian Nuswantoro

Sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro merupakan sistem yang digunakan calon mahasiswa Universitas Dian Nuswantoro untuk mendaftar dan menjadi mahasiswa, sistem ini masih menggunakan *monolithic architecture* dimana arsitektur ini memiliki kekurangan yang cukup krusial. Dengan ini penulis menawarkan untuk berpindah dari *monolithic architecture* menjadi *multi tier architecture*. *Multi tier architecture* merupakan sebuah arsitektur sistem dimana sebuah aplikasi biasanya dibagi menjadi dua bagian, *server-side* dan *client-side*. *Server-side* terdiri dari kode pemanggilan basis data, logika, dan kalkulasi sedangkan *client-side* berisi tampilan atau UI dari sistem.

*Server-side* dan *client-side* merupakan dua buah aplikasi yang berbeda tetapi saling berkomunikasi. Cara mereka berkomunikasi ialah menggunakan protokol HTTP. Saat ini aplikasi yang menggunakan arsitektur ini kebanyakan menggunakan REST untuk metode komunikasinya. REST merupakan metode yang sangat populer untuk saat ini. Cara komunikasi antar keduanya ialah dari *server-side* nantinya akan menyediakan url yang nantinya akan dipanggil di *client-side*.

REST memiliki banyak jenis url, GET dan POST merupakan jenis url yang sering digunakan para pengembang. Url ini nantinya dapat diisi dengan variabel yang bisa disisipkan di url-nya secara langsung, melalui body url dan juga melalui header url. Respon yang diterima pihak *client-side* ialah data yang berformat JSON, aplikasi *client-side* lah yang nantinya melakukan parsing data dari respon yang diterima menjadi data yang siap

ditampilkan di UI. Dengan arsitektur seperti ini masalah masalah yang ada pada *monolithic architecture* dapat teratasi.

#### 4.2.2 GraphQL

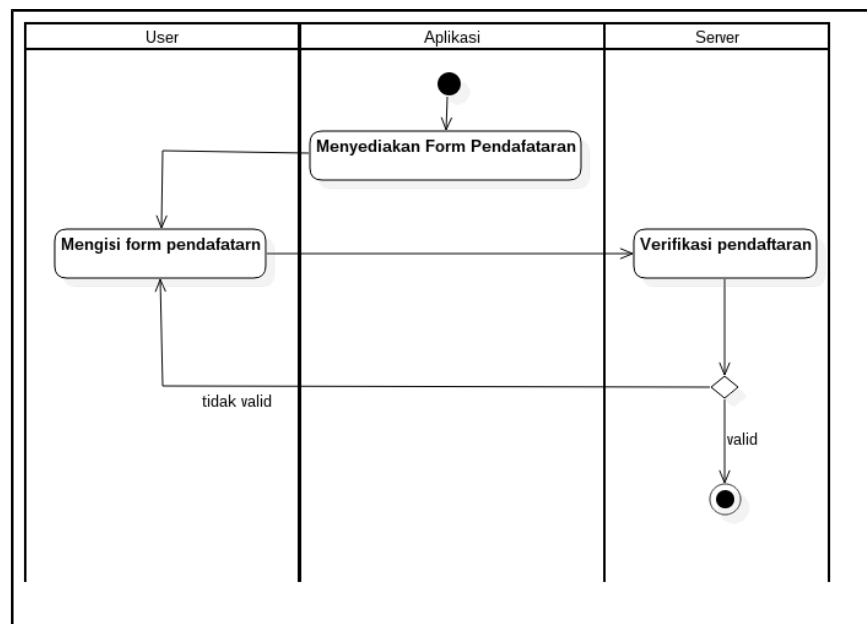
GraphQL merupakan salah satu metode komunikasi *server-side* dan *client-side*, tetapi konsep dari GraphQL berbeda dengan REST dimana data yang dikirim ke *client-side* berformat JSON yang mana struktur data dari REST bersifat statik sedangkan struktur data yang dikirim GraphQL bersifat dinamis, jadi *client-side* dapat meminta data yang diperlukan saja tanpa boros bandwidth yang diperlukan untuk proses tranfer data dan juga dapat mempercepat permintaan. Perbedaan juga terdapat pada cara kerja komunikasinya, *server-side* menyediakan fungsi yang *client-side* gunakan untuk meminta data. Dari fungsi itulah yang nantinya akan diterjemahkan menjadi sebuah permintaan HTTP agar kedua pihak dapat berkomunikasi. Dengan menggunakan metode ini *client-side* tidak perlu lagi melakukan parsing data karena data yang dikirimkan sudah data yang siap digunakan.

### 4.3 Rancangan Sistem

#### 4.3.1 Perancangan Activity Diagram

Berikut merupakan diagram aktivitas dari sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro

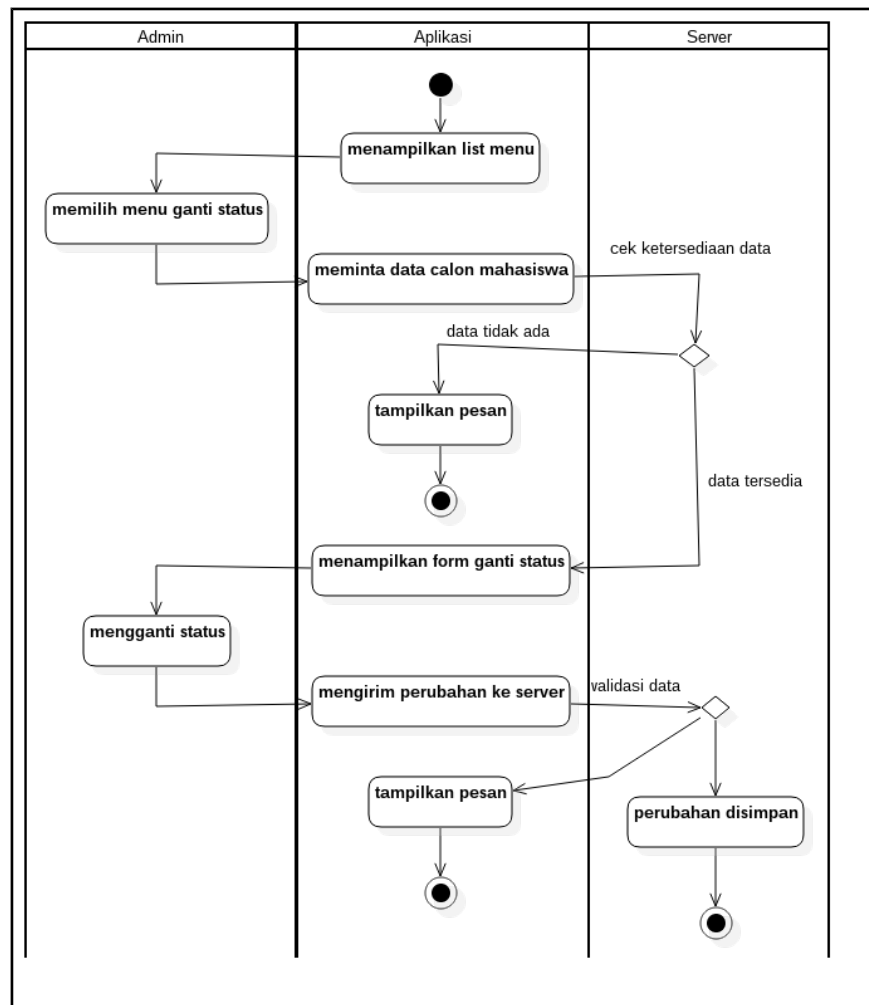
##### 1. Activity Diagram Pendaftaran



Gambar 4.1 Activity Diagram Pendaftaran

Diagram aktivitas ini menggambarkan aktivitas user ketika melakukan pendaftaran di sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro. Pengisian data pada form perndaftaran akan ditolak atau tidak valid ketika user melakukan kesalahan pada inputan.

## 2. Activity Diagram ganti status calon mahasiswa

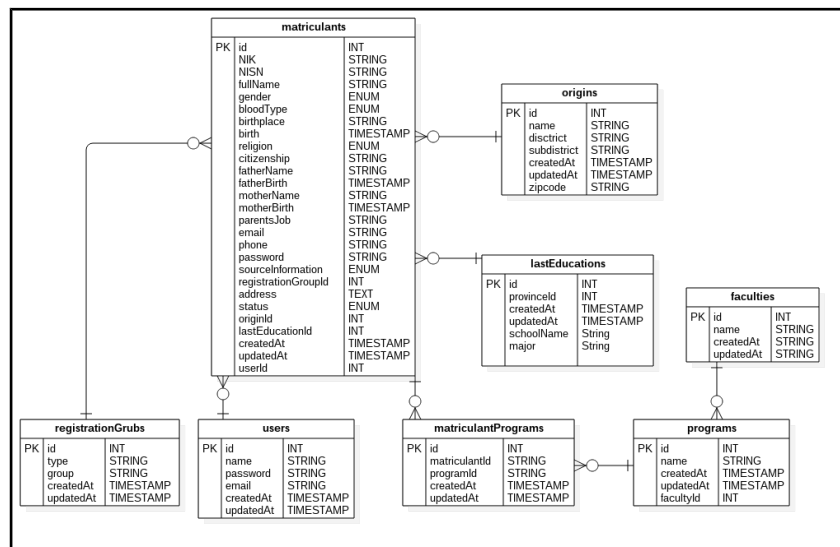


Gambar 4.2 Activity Diagram ganti status calon mahasiswa

Activity diagram diatas menggambarkan aktivitas admin ketika mengganti status calon mahasiswa. Sebelum melakukan penggantian status sistem akan melakukan pencarian data calon mahasiswa yang akan diganti, ketika data ada dan valid maka akan muncul form untuk mengganti status calon mahasiswa tersebut.

### 4.3.2 Perancangan Basis Data

Berikut merupakan skema basis data dari sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro berbasis *backend API GraphQL web service* berupa ERD.



Gambar 4.3 ERD basis data sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro

Gambar diatas merupakan rancangan ERD dari sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro berbasis GraphQL *web service*. Untuk mengurangi redundansi data pada sistem ini penulis melakukan pemecahan tabel dan juga memberikan relasi pada setiap pecahan tabel yang berhubungan. Berikut merupakan penjelasan dari tabel-tabel yang dipecah oleh penulis beserta dengan relasinya.

## 1. Tabel matriculants

*Tabel 4.1 Tabel matriculants*

| No | Nama Field          | Tipe Data | Keterangan                               |
|----|---------------------|-----------|--|
| 1  | id                  | Integer   | Primary key, auto increment              |
| 2  | NIK                 | Varchar   | Nomor identitas kependudukan             |
| 3  | NISN                | Varchar   | Nomor identitas sekolah                  |
| 4  | fullName            | Varchar   | Nama user                                |
| 5  | gender              | Enum      | Jenis kelamin user                       |
| 6  | bloodType           | Enum      | Golongan darah user                      |
| 7  | birthPlace          | Varchar   | Tempat lahir user                        |
| 8  | birth               | Timestamp | Tanggal lahir user                       |
| 9  | religion            | Enum      | Agama user                               |
| 10 | fitizenship         | Varchar   | Kewarganegaraan user                     |
| 11 | fatherName          | Varchar   | Nama ayah user                           |
| 12 | fatherBirth         | Timestamp | Tanggal lahir ayah user                  |
| 13 | motherName          | Varchar   | Nama ibu user                            |
| 14 | motherBirth         | Timestamp | Tanggal lahir ibu user                   |
| 15 | parentsJob          | Varchar   | Pekerjaan orang tua user                 |
| 16 | email               | Varchar   | Email user                               |
| 17 | phone               | Varchar   | Nomor telfon user atau wali              |
| 18 | password            | Varchar   | Password user untuk mencetak kartu ujian |
| 19 | sourceInformation   | Enum      | Sumber info userssss                     |
| 20 | registrationGroupId | Integer   | Jenis kelompok pendaftaran user          |
| 21 | address             | Varchar   | Alamat user                              |
| 22 | status              | Enum      | Status user                              |
| 23 | originId            | Integer   | Id asal user                             |
| 24 | lastEducationId     | Integer   | Id pendidikan terakhir user              |

|    |           |           |                            |
|----|-----------|-----------|----------------------------|
| 25 | createdAt | timestamp | Tanggal user daftar        |
| 26 | updatedAt | timestamp | Tanggal data user diupdate |
| 27 | userId    | integer   | Id user                    |

Tabel ini merupakan tabel untuk menampung data calon mahasiswa yang mendaftar. Tabel ini memiliki relasi dengan tabel lain, berelasi dengan tabel user dimana banyak matriculant akan divalidasi oleh satu user, berelasi dengan tabel registrationGroup dimana berfungsi untuk mengelompokkan jenis pendaftaran dari setiap calon mahasiswa yang mendaftar jenis relasinya setiap jenis pendaftaran akan memiliki banyak calon mahasiswa, berelasi dengan tabel origin dimana setiap tempat tinggal memiliki banyak calon mahasiswa, berelasi dengan tabel lastEducation yang bersifat *one-to-many* dari tabel lastEducation dimana setiap sekolah asal memiliki banyak calon mahasiswa yang mendaftar, dan yang terakhir berelasi dengan tabel program dimana jenis relasinya ialah *many-to-many* setiap calon mahasiswa dapat memilih dua jurusan dan setiap jurusan memiliki banyak calon mahasiswa dan tabel pivot dari relas tersebut ialah tabel matriculantProgram. Isi tabel matriculantProgram merupakan programId dan matriculantId dimana keduanya adalah *primary key* dari tabel program dan tabel matriculant.

## 2. Tabel registrationGroups

*Tabel 4.2 Tabel registrationGroups*

| No | Nama Field | Tipe Data | Keterangan                  |
|----|------------|-----------|-----------------------------|
| 1  | id         | integer   | Primary key, auto increment |
| 2  | type       | varchar   | Tipe pendaftaran            |
| 3  | group      | varchar   | Grub pendaftaran            |
| 4  | createdAt  | timestamp | Tanggal data dibuat         |



|   |           |           |                              |
|---|-----------|-----------|------------------------------|
| 5 | updatedAt | timestamp | Tanggal data terakhir diedit |
|---|-----------|-----------|------------------------------|

Tabel registrationGroup hanya memiliki satu relasi menuju tabel matriculant dimana jenis relasinya ialah *one-to-many*, jadi setiap jenis pendaftaran memiliki banyak calon mahasiswa.

### 3. Tabel lastEducations

*Tabel 4.3 Tabel lastEducations*

| No | Nama Field | Tipe Data | Keterangan                           |
|----|------------|-----------|--------------------------------------|
| 1  | id         | integer   | Primary key, auto increment          |
| 2  | schoolName | varchar   | Nama sekolah sebelumnya              |
| 3  | major      | varchar   | Jurusan yang diambil user sebelumnya |
| 4  | originId   | integer   | Id origin user                       |
| 5  | createdAt  | timestamp | Tanggal data dibuat                  |
| 6  | updatedAt  | timestamp | Tanggal data diedit                  |

Tabel lastEducation ini memiliki relasi hanya ke tabel matriculant dimana berjenis *one-to-many* yang mana setiap jenis pendaftaran dapat memiliki banyak calon mahasiswa.

### 4. Tabel origins

*Tabel 4.4 Tabel origins*

| No | Nama Field  | Tipe Data | Keterangan                  |
|----|-------------|-----------|-----------------------------|
| 1  | id          | integer   | Primary key, auto increment |
| 2  | name        | varchar   | Nama daerah asal            |
| 3  | district    | varchar   | Nama kabupaten asal         |
| 4  | subDistrict | varchar   | Nama kecamatan asal         |

|   |           |           |                     |
|---|-----------|-----------|---------------------|
| 5 | zipcode   | varchar   | Kode pos            |
| 6 | createdAt | timestamp | Tanggal data dibuat |
| 7 | updatedAt | timestamp | Tanggal data diedit |

Tabel origin memiliki satu relasi dengan tabel matriculant dimana jenis relasinya ialah *one-to-many* , jadi setiap asal calon mahasiswa memiliki banyak calon mahasiswa yang mendaftar.

#### 5. Tabel matriculantPrograms

*Tabel 4.5 Tabel matriculantPrograms*

| No | Nama Field    | Tipe Data | Keterangan                  |
|----|---------------|-----------|-----------------------------|
| 1  | id            | integer   | Primary key, auto increment |
| 2  | matriculantId | integer   | Id matriculant              |
| 3  | programId     | integer   | Id major                    |
| 4  | createdAt     | timestamp | Tanggal data dibuat         |
| 5  | updatedAt     | timestamp | Tanggal data diedit         |

Tabel matriculantPrograms merupakan tabel pivot yang menghubungkan relasi *many-to-many* dari tabel matriculant ke tabel program. Setiap calon mahasiswa dapat memilih dua jurusan dan setiap jurusan memiliki banyak calon mahasiswa.

#### 6. Tabel programs

*Tabel 4.6 Tabel programs*

| No | Nama Field | Tipe Data | Keterangan                  |
|----|------------|-----------|-----------------------------|
| 1  | id         | integer   | Primary key, auto increment |
| 2  | name       | varchar   | Nama jurusan                |
| 3  | facultyId  | integer   | Id fakultas                 |

|   |           |           |                     |
|---|-----------|-----------|---------------------|
| 4 | createdAt | timestamp | Tanggal data dibuat |
| 5 | updatedAt | timestamp | Tanggal data diedit |

Tabel ini memiliki dua buah relasi ke tabel lain, pertama berelasi *one-to-many* dengan tabel faculties jadi banyak jurusan akan ditampung dalam satu fakultas dan berselasi *many-to-many* dengan tabel matriculant yang mana tabel matriculantProgram menjadi pivot atau tabel penghubung.

#### 7. Tabel faculties

*Tabel 4.7 Tabel faculties*

| No | Nama Field | Tipe Data | Keterangan                  |
|----|------------|-----------|-----------------------------|
| 1  | id         | integer   | Primary key, auto increment |
| 2  | name       | varchar   | Nama fakultas               |
| 3  | createdAt  | timestamp | Tanggal data dibuat         |
| 4  | updatedAt  | timestamp | Tanggal data diedit         |

Tabel faculties memiliki memiliki satu relasi ke tabel program, dimana jenis relasinya ialah *one-to-many*, jadi setiap fakultas memiliki banyak jurusan.

#### 8. Tabel users

*Tabel 4.8 Tabel users*

| No | Nama Field | Tipe Data | Keterangan                  |
|----|------------|-----------|-----------------------------|
| 1  | id         | integer   | Primary key, auto increment |
| 2  | name       | varchar   | Nama dari user              |
| 3  | password   | varchar   | Password user               |
| 4  | email      | varchar   | Email user                  |
| 5  | createdAt  | timestamp | Tanggal data dibuat         |

|   |           |           |                     |
|---|-----------|-----------|---------------------|
| 6 | updatedAt | timestamp | Tanggal data diedit |
|---|-----------|-----------|---------------------|

Tabel user memiliki satu relasi menuju ke tabel matriculant yang berjenis *one-to-many* dimana setiap user dapat melakukan validasi ke banyak calon mahasiswa.

#### 4.3.3 Kueri

Dari perancangan ERD dipembahasan sebelumnya, penulis dapat merancang kueri yang nantinya akan diterjemahkan ke dalam kueri graphql. Kuery GraphQL ini yang nanti akan diimplementasikan kedalam Backend API GraphQL. Berikut merupakan rancangan kueri SQL:

##### 1. Query SQL daftar matriculant baru

Query SQL dibawah ini digunakan untuk menambah data calon mahasiswa baru Universitas Dian Nuswantoro. Respsion yang didapatkan dari query SQL ini merupakan data yang berbentuk *row object* yang setiap atributnya tidak bisa dikurangi atau dikustomisasi sesuai kebutuhan sistem.

```

INSERT INTO public."Matriculants" ("NIK","NISN","fullName",
                                   "gender","bloodType","birthPlace",
                                   birth,religion,citizenship,"fatherName",
                                   "fatherBirth","motherName","motherBirth",
                                   "parentsJob","email","phone","password",
                                   "sourceInformation","registrationGroupId",
                                   "address","status","originId",
                                   "lastEducationId","createdAt","updatedAt")
VALUES ('009929389928','678799dy88','Jhon Doe','MALE','A',
        'somewhere','2018-05-30 23:58:39','ISLAM','Indonesia',
        'foo bar','2018-05-30 23:58:39','bar foo',
        '2018-05-30 23:58:39','string','jhondoe@gmail.com',
        '09231231','secret','TEMAN',1,'somewhere',
        'Ujian',1,3,'2018-06-08 02:46:18','2018-06-08 02:46:18');

```

Gambar 4.4: Query SQL daftar matriculant baru

2. Query SQL melihat data calon mahasiswa yang berhasil mendaftar

Dibawah ini merupakan query SQL yang menampilkan data calon mahasiswa beserta data asal tempat dan juga asal sekolah. Untuk mengambil data tersebut penulis harus menggunakan query untuk menggabungkan tabel yang sudah dipecah karena data yang diambil berada di tiga tabel berbeda.

```

select * from "Matriculants"
left join "MatriculantPrograms"
on "MatriculantPrograms"."matriculantId" = "Matriculants".id
left join "LastEducations"
on "LastEducations"."id" = "Matriculants"."lastEducationId"
left join "Origins"
on "Origins"."id" = "Matriculants"."originId";

```

Gambar 4.5: Query SQL melihat data calon mahasiswa

### 3. Query SQL mengganti status calon mahasiswa

Query dibawah digunakan untuk mengganti status calon mahasiswa. Yang dapat mengganti status calon mahasiswa ialah admin, dimana ada empat jenis status dari calon mahasiswa, jenis-jenisnya ialah Daftar, Registrasi, Ujian, Mundur.

```
UPDATE "Matriculants" SET status="Mundur" WHERE id=26;
```

Gambar 4.6: Query SQL mengganti status calon mahasiswa

### 4. Query SQL menampilkan data pendaftar berdasarkan jurusan

Query dibawah ini digunakan untuk menampilkan data calon mahasiswa berdasarkan jurusan yang diambil. Untuk mendapatkan data tersebut memerlukan penggabungan tabel matriculant, matriculantProgram, dan program dengan query, karena data yang diperlukan berada ditiga tabel tersebut.

```
select * from "Matriculants"  
inner join "MatriculantPrograms" on "Matriculants"."id" =  
"MatriculantPrograms"."matriculantId"  
inner join "Programs" on "MatriculantPrograms"."programId"="Programs"."id"  
where "Programs"."id"=5;
```

Gambar 4.7: Query SQL menampilkan data pendaftar berdasarkan jurusan

5. Query SQL menampilkan daftar calon mahasiswa yang mundur berdasarkan jurusan

Query dibawah ini digunakan untuk menampilkan daftar calon mahasiswa berdasarkan jurusan dan statusnya. Untuk mendapatkan data tersebut query yang digunakan harus menggabungkan tiga tabel, tabel matriculant, matriculantProgram, dan program , karena data yang diminta terdapat di tabel tabel tersebut.

```
select count(*) from "Matriculants"  
inner join "MatriculantPrograms" on "Matriculants"."id" =  
"MatriculantPrograms"."matriculantId"  
inner join "Programs" on "MatriculantPrograms"."programId"="Programs"."id"  
where "Matriculants"."status"='Daftar' and "Programs"."id"=5;
```

Gambar 4.8: Query SQL menampilkan daftar calon mahasiswa yang mundur berdasarkan jurusan

6. Query SQL menampilkan daftar calon mahasiswa berdasarkan jurusan dan sekolah asal

```
select * from "Matriculants"  
inner join "LastEducations" on  
"Matriculants"."lastEducationId"="LastEducations"."id"  
inner join "MatriculantPrograms" on "Matriculants"."id" =  
"MatriculantPrograms"."matriculantId"  
inner join "Programs" on "MatriculantPrograms"."programId"="Programs"."id"  
where "Matriculants"."lastEducationId" = 5 and "Programs"."id"=5;
```

Gambar 4.9: Query SQL menampilkan daftar calon mahasiswa berdasarkan jurusan dan sekolah asal

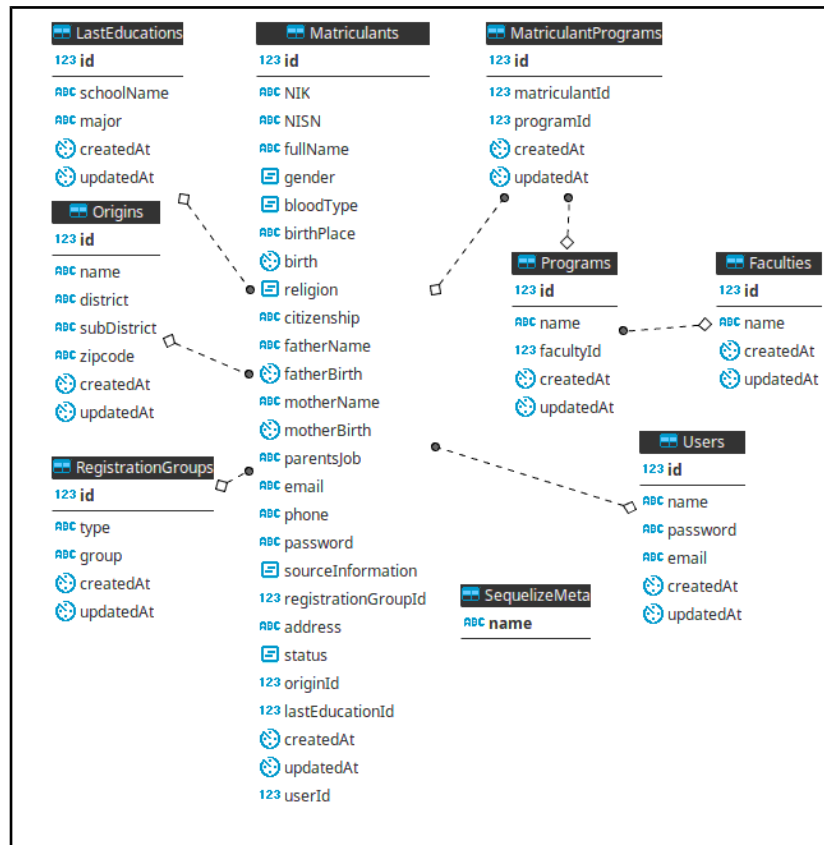
Untuk mendapatkan data tersebut, query harus menggabungkan empat tabel, tabel matriculant, tabel lastEducation, tabel matriculantProgram, tabel program, karena data yang diminta terdapat pada empat tabel tersebut.

## **4.4 Implementasi Sistem**

### **4.4.1 Implementasi Basis Data**

Berikut adalah hasil implementasi basis data dari sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro berbasis Backend API GraphQL *webservice*





Gambar 4.10: Skema basis data sistem PMB Universitas Dian Nuswantoro

#### 4.4.2 Implementasi GraphQL

Berikut adalah implementasi GraphQL pada sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro dari rancangan sistem yang telah dibahas di pembahasan sebelumnya.

1. Hasil GraphQL menyimpan data calon mahasiswa

```
mutation createMatriculant {
  createMatriculant(input:
    {NIK: "009929389928",
      NISN: "678799dyy88",
      fullName: "Jhon Doe",
      gender: MALE,
      bloodType: A,
      birthPlace: "somewhere",
      birth: "Wed May 30 2018 23:58:39 GMT+0700 (WIB)",
      religion: ISLAM,
      citizenship: "Indonesia",
      fatherName: "foo bar",
      fatherBirth: "Wed May 30 2018 23:58:39 GMT+0700 (WIB)",
      motherName: "bar foo",
      motherBirth: "Wed May 30 2018 23:58:39 GMT+0700 (WIB)",
      parentsJob: "string",
      email: "jhondoe@gmail.com",
      phone: "09231231",
      password: "secret",
      sourceInformation: TEMAN,
      RegistrationGroup: 2,
      address: "somewhere",
      status: Ujian,
      Origin: 8,
      LastEducation: 5,
      userId: 2,
      majorOne: 5,
      majorTwo: 6}) {
    id
  }
}
```

Gambar 4.11 Mutation GraphQL menyimpan data user baru

Mutation GraphQL diatas merupakan *function* untuk menyimpan data calon mahasiswa baru, dimana fungsi tersebut

akan memberikan respon atribut NIK, NISN, fullName, gender, bloodType setelah sukses menyimpan data. Respon data bisa diganti sesuai kebutuhan.

2. Hasil GraphQL melihat data calon mahasiswa yang berhasil mendaftar

```
query findMatriculant {  
  matriculant(id: 90) {  
    id  
    fullName  
    NIK  
    NISN  
    bloodType  
    Programs{  
      name  
    }  
  }  
}
```

Gambar 4.12 Query GraphQL menampilkan data user

Query GraphQL diatas digunakan menampilkan data calon mahasiswa yang memiliki id 90, dimana dalam query tersebut penulis hanya meminta atribut id, nama, golongan darah, NIK, NISN dan jurusan calon mahasiswa yang berhasil mendaftar. Dari *query* graphql diatas penulis harus menghubungkan tabel tabel yang sudah dipecah karena data tersebut berada di beberpa tabel Atribut yang diminta melalui fungsi graphql sangatlah fleksibel tergantung dari kebutuhan dari *client-side*.

3. Hasil GraphQL mengganti status calon mahasiswa Universitas Dian Nuswantoro

```
mutation changeStatus{
  changeStatusMatriculant(input:{
    id:90,
    status:Ujian
  }){
    id
    fullName
    status
  }
}
```

Gambar 4.13 Mutation GraphQL mengganti status user

Mutation GraphQL diatas merupakan *function* untuk mengganti status calon mahasiswa. Tipe status mahasiswa sendiri ada empat daftar, registrasi, mundur, dan ujian. *Function* ini akan digunakan ketika calon mahasiswa selesai menyelesaikan suatu tahap dalam proses penerimaan mahasiswa baru Universitas Dian Nuswantoro. Jika sesuai *function* diatas berhasil mengganti status dari calon mahasiswa Universitas Dian Nuswantoro maka *function* akan menampilkan respon data id, fullName, dan status. Respon data bisa diganti sesuai kebutuhan *client-side*.

4. Hasil GraphQL melakukan pencarian calon mahasiswa

```
query statMatriculant {  
  matriculantStatistic(status:Daftar,date:"2017") {  
    id  
    fullName  
    status  
    Programs{  
      name  
    }  
  }  
}
```

Gambar 4.14 Query GraphQL pencarian seluruh calon mahasiswa

Query GraphQL diatas digunakan untuk menampilkan data calon mahasiswa yang bisa disaring dengan paling sedikit satu parameter dan paling banyak empat parameter. Empat parameter ini berupa date, status, tipe pendaftaran, dan juga sekolah asal. Parameter-parameter pada query ini bersifat fleksibel dimana *client-side* dapat menggunakan satu parameter, dua parameter, tiga parameter dan paling banyak empat parameter. Hal ini bertujuan agar *function* graphql ini dapat digunakan berkali kali sesuai dengan kebutuhan *client-side*. Jika sesuai dengan query diatas *function* akan memberikan respon data berupa id, fullName, status. Respon data bisa diganti sesuai kebutuhan *client-side*.

5. Hasil GraphQL menampilkan jumlah pendaftar berdasarkan tahun dan bulan

```
query matriculantPerMonth{  
  matriculantPerMonth(year:2018){  
    jan  
    feb  
    mar  
    apr  
    may  
    jun  
    jul  
    ags  
    sep  
    oct  
    nov  
    dec  
  }  
}
```

Gambar 4.15 Query GraphQL jumlah calon mahasiswa berdasarkan tahun dan bulan

Query GraphQL diatas digunakan untuk menampilkan data statistik pertumbuhan calon mahasiswa yang mendaftar berdasarkan tahun dan bulan daftarnya. Respon dari *function* ini merupakan jumlah data calson mahasiswa yang mendaftar berdasarkan bulan dan tahun sebagai parameter penyaringnya. Tiap atribut yang dikirimkan melalui respon yang sukses merupakan jumlah dari pertumbuhan pendaftar pada sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro dan atribut respon bisa diambil sesuai dengan kebutuhan *client-side*.

6. Hasil GraphQL menampilkan jumlah pendaftar berdasarkan tipe pendaftaran

```
query sortMatriculant {  
  sortMatriculant(type: "reguler") {  
    ia  
    ib  
    ic  
    iia  
    iib  
  }  
}
```

Gambar 4.16 Query GraphQL menampilkan jumlah calon mahasiswa berdasarkan tipe pendaftaran

Query GraphQL diatas digunakan untuk menampilkan jumlah data calon mahasiswa berdasarkan tipe pendaftarannya dan akan menampilkan respon data berupa gelombang pendaftaran dari tipe yang sudah di filter sebelumnya. Untuk menjadikan query ini memerlukan tabel matriculant dan tabel registrationGroup karena data yang diperlukan terdapat pada dua tabel tersebut.

7. Hasil GraphQL menampilkan daftar calon mahasiswa berdasarkan status dan jurusan

```
query MatriculantByProgram{  
  MatriculantByProgram(programId:5,status:"Daftar"){  
    id  
    fullName  
    status  
    Programs{  
      name  
    }  
  }  
}
```

Gambar 4.17 Hasil GraphQL menampilkan daftar calon mahasiswa berdasarkan status dan jurusan

Query GraphQL diatas digunakan untuk menampilkan daftar calon mahasiswa mendaftar berdasarkan status dan jurusan yang diambil. Respon yang akan diberikan merupakan daftar calon mahasiswa yang masuk dalam kategori.



8. Hasil GraphQL menampilkan daftar calon mahasiswa berdasarkan asal sekolah dan jurusan

```
query MatriculantByLastEdu{
  MatriculantByLastEdu(programId:5,lastEduId:5){
    email
    fullName
    LastEducation{
      schoolName
    }
    Programs{
      name
    }
  }
}
```

Gambar 4.18 Hasil GraphQL menampilkan daftar calon mahasiswa berdasarkan asal sekolah dan jurusan

Query GraphQL diatas digunakan untuk menampilkan daftar calon mahasiswa yang mendaftar berdasarkan asal sekolah dan jurusan yang diambil.

## BAB V

### HASIL PENELITIAN DAN PEMBAHASAN

#### 5.1 Hasil Penelitian

##### 5.1.1 Hasil Implementasi Backend

Dalam pembahasan ini penulis akan memaparkan bentuk *request*, *response*, *developer tool* dari sistem yang sudah dikembangkan.

##### 1. *Developer tool*



Gambar 5.1 Developer tool graphQL

Gambar diatas merupakan *developer tool* dimana halaman ini digunakan untuk pengembang melakukan *debuging* sekaligus dokumentasi para pengembang. Halaman ini secara otomatis dibuat ketika melakukan *initial project*.

## 2. Fungsi createMatriculant

```
mutation createMatriculant {
  createMatriculant(input:
    {NIK: "009929389928", NISN: "678799dyy88",
      fullName: "Jhon Doe", gender: MALE,
      bloodType: A, birthPlace: "somewhere",
      birth: "Wed May 30 2018 23:58:39 GMT+0700 (WIB)",
      religion: ISLAM, citizenship: "Indonesia",
      fatherName: "foo bar", fatherBirth: "Wed May 30 2018
        23:58:39 GMT+0700 (WIB)",
      motherName: "bar foo", motherBirth: "Wed May 30 2018
        23:58:39 GMT+0700(WIB)",
      parentsJob: "string", email: "jhondoe@gmail.com",
      phone: "09231231", password: "secret",
      sourceInformation: TEMAN, RegistrationGroup: 2,
      address: "somewhere", status: Ujian,
      Origin: 8, LastEducation: 5,
      userId: 2, majorOne: 5,
      majorTwo: 6})) {
    id
  }
}
```

Gambar 5.2 Request mutation createMatriculant

```
{
  "data": {
    "matriculant": {
      "id": 90,
    }
  }
}
```

Gambar 5.3 Response mutation  
createMatriculant

Gambar diatas merupakan bentuk *request* dan *response* yang digunakan untuk menambah data calon mahasiswa. *Request* diatas bertipe *mutation* karena bersifat mengubah atau menambah data. *Response* yang diterima setelah *request* berhasil berupa id dari data yang berhasil disimpan, sebenarnya semua atribut dari data yang sudah disimpan dapat di *retrieve* tergantung kepada kebutuhan sistem.

### 3. Fungsi findMatriculant

```
query findMatriculant {
  matriculant(id: 90) {
    id fullName
    gender birth
    birthPlace password
    parentsJob phone
    status userId
    bloodType religion
    fatherName motherName
    Programs{
      name
    }
  }
}
```

Gambar 5.4 Request fungsi findMatrulant

```

{
  "data": {
    "matriculant": {
      "id": 90,
      "fullName": "Darwin Bode Jr.",
      "gender": "FEMALE",
      "birth": "Thu Jun 28 2018 03:19:27 GMT+0700 (WIB)",
      "birthPlace": "Berkshire",
      "password": "WXElLvGKcPfg9l5",
      "parentsJob": "National Data Orchestrator",
      "phone": "+624598024",
      "status": "Ujian",
      "userId": 4,
      "bloodType": "O",
      "religion": "BUDHA",
      "fatherName": "Bertha Altenwerth",
      "motherName": "Rosalia Bernhard MD",
      "Programs": [{"name": "Desain Komunikasi Visual S1"},
        {"name": "Magister Manajemen S1"}]
    }
  }
}

```

Gambar 5.5 Response fungsi findMatrulant

Gambar diatas merupakan bentuk *request* dan *response* dari fungsi findMatriculant. Fungsi ini digunakan untuk mencari data calon mahasiswa berdasarkan id. Fungsi ini juga bertipe query karena hanya bersifat menampilkan data. Untuk menampilkan data dari tabel relasi hanya perlu menambahkan attribut yang berelasi dengan format seperti atribut Programs dari *request* diatas dan *response*-nya akan berbentuk seperti gambar diatas.

#### 4. Fungsi changeStatusMatriculant

```
mutation changeStatus{
  changeStatusMatriculant(input:{
    id:90,
    status:Ujian
  }){
    id
    fullName
    status
  }
}
```

Gambar 5.6 Reqeust mutation  
changeStatusMatriculant

```
{
  "data": {
    "changeStatusMatriculant": {
      "id": 90,
      "fullName": "Darwin Bode Jr.",
      "status": "Ujian"
    }
  }
}
```

Gambar 5.7 Responsemutation  
changeStatusMatriculant

Gambar diatas merupakan *request* dan *response* dari fungsi changeStatusMatriculant. Fungsi ini digunakan untuk mengganti status calon mahasiswa. Fungsi ini memiliki dua parameter yang bersifat wajib, parameter pertama ialah id calon mahasiswa yang ingin diganti dan parameter kedua adalah status, untuk mengganti

statusnya. *Respons* yang diterima merupakan data calon mahasiswa yang sudah diganti statusnya.

5. Fungsi statMatriculant

```
query statMatriculant {  
  matriculantStatistic(status:Daftar,date:"2017") {  
    fullName  
    status  
  }  
}
```

Gambar 5.8 Request query statMatriculant

```
{  
  "data": {  
    "matriculantStatistic": [  
      {  
        "fullName": "Edd Lowe",  
        "status": "Daftar"  
      },  
      {  
        "fullName": "Savanah Doyle",  
        "status": "Daftar"  
      },  
      {  
        "fullName": "Justen Turner",  
        "status": "Daftar"  
      }  
    ]  
  }  
}
```

Gambar 5.9 Response query statMatriculant

Gambar diatas merupakan fungsi graphql statMatriculant bersifat query. Fungsi ini digunakan untuk mencari data calon mahasiswa berdasarkan empat kategori, tahun daftar, asal sekolah, jenis pendaftaran dan status calon mahasiswa. *Response* yang diterima merupakan *array of object* dari data calon mahasiswa yang masuk dalam kategori.

6. Fungsi matriculantPerMonth

```
query matriculantPerMonth{  
  matriculantPerMonth(year:2018){  
    jan  
    feb  
    mar  
    apr  
    may  
    jun  
    jul  
    ags  
    sep  
    oct  
    nov  
    dec  
  }  
}
```

Gambar 5.10 Request query matriculantPerMonth



```
{
  "data": {
    "matriculantPerMonth": {
      "jan": 8,
      "feb": 8,
      "mar": 12,
      "apr": 9,
      "may": 6,
      "jun": 7,
      "jul": 12,
      "ags": 0,
      "sep": 0,
      "oct": 0,
      "nov": 0,
      "dec": 0
    }
  }
}
```

Gambar 5.11 Responsequery matriculantPerMonth

Gambar diatas merupakan bentuk *request* dan *response* dari fungsi `matriculantPerMonth` dimana fungsi ini berfungsi untuk menghitung jumlah calon mahasiswa berdasarkan bulan dan tahun daftarnya sebagai paramater wajib. *Response* dari fungsi ini bisa dilihat dari gambar sebelah kanan atas.

## 7. Fungsi sortMatriculant

```
query sortMatriculant {  
  sortMatriculant(type: "reguler") {  
    ia  
    ib  
    ic  
    iia  
    iib  
  }  
}
```

Gambar 5.12 Request query sortMatriculant

```
{  
  "data": {  
    "sortMatriculant": {  
      "ia": 23,  
      "ib": 16,  
      "ic": 17,  
      "iia": 16,  
      "iib": 15  
    }  
  }  
}
```

Gambar 5.13 Response query sortMatriculant

Fungsi diatas digunakan untuk mengitung dan mengelompokan jumlah calon mahasiswa berdasarkan tipe pendaftarannya, fungsi ini bersifat query karena tidak mengubah atau menambah data. Tipe pendaftaran menjadi parameter dari fungsi ini.

8. Fungsi matriculantByProgram

```
query MatriculantByProgram{
  MatriculantByProgram(programId:5,status:"Daftar"){
    id
    fullName
    status
    Programs{
      name
    }
  }
}
```

Gambar 5.14 Request dari fungsi  
matriculantByProgram

```
{
  "data": {
    "MatriculantByProgram": [
      {
        "id": 58,
        "fullName": "Abbigail Dietrich DDS",
        "status": "Daftar",
        "Programs": [{"name": "Teknik Informatika S1"}]
      },
      {
        "id": 67,
        "fullName": "Justen Turner",
        "status": "Daftar",
        "Programs": [{"name": "Teknik Informatika S1"}]
      }
    ]
  }
}
```

Gambar 5.15 Response dari fungsi matriculantByProgram

Fungsi diatas digunakan untuk menyaring list data calon mahasiswa berdasarkan jurusan yang diambil dan status dari calon mahasiswa tersebut. Ada dua parameter wajib dari fungsi ini pertama `programId` yang merupakan id dari program jurusan dan status merupakan status dari calon mahasiswa tersebut. *Response* dari fungsi ini berupa *array of object* dari data calon mahasiswa yang terkategori.

9. Fungsi `matriculantByLastEdu`

```
query MatriculantByLastEdu{
  MatriculantByLastEdu(programId:5,lastEduId:5){
    email
    fullName
    LastEducation{
      schoolName
    }
    Programs{
      name
    }
  }
}
```

Gambar 5.16 Request fungsi query `matriculantByLastEdu`

```
{
  "data": {
    "MatriculantByProgram": [
      {
        "id": 58,
        "fullName": "Abbigail Dietrich DDS",
        "status": "Daftar",
        "Programs": [{"name": "Teknik Informatika S1"}]
      },
      {
        "id": 67,
        "fullName": "Justen Turner",
        "status": "Daftar",
        "Programs": [{"name": "Teknik Informatika S1"}]
      }
    ],
  },
}
```

Gambar 5.17 Response fungsi query matriculantByLastEdu

Fungsi diatas digunakan untuk menampilkan list data calon mahasiswa berdasarkan jurusan yang diambil dan asal sekolah calon mahasiswa. Ada dua parameter yang bersifat wajib dari fungsi diatas `programId` yang merupakan id dari jurusan yang diambil dan `lastEducationId` merupakan id dari asal sekolah calon mahasiswa yang mendaftar. *Response* dari fungsi ini berupa *array of object* dari data calon mahasiswa yang telah terkategori.

## 5.1.2 Tampilan Aplikasi

### 1. Halaman Form Pendaftaran

**FORM PENDAFTARAN MAHASISWA**

Penitia Penerimaan Mahasiswa Baru Udinus mengundang pada seluruh calon mahasiswa untuk satu website dan berhati-hati terhadap segala jenis penipuan yang mengatasnamakan Panitia Penerimaan Mahasiswa Baru Udinus. Untuk pembayaran transfer menggunakan nomor rekening atas nama Universitas Dian Nuswantoro (bukan rekening an pribadi) dan Panitia tidak memungut biaya lain selain yang diinformasikan melalui surat resmi. Selain itu segala kegiatan dan informasi tentang Penerimaan Mahasiswa Baru Udinus akan selalu di publikasikan di website resmi [www.dinus.ac.id](http://www.dinus.ac.id). **Terima kasih**

Jika Anda sudah pernah melakukan pendaftaran namun tidak bisa login, jangan melakukan pendaftaran kembali (berkali-kali). Hubungi customer service kami jika Anda menemui kesulitan login. Data pendaftaran yang kembar akan kami hapus.

Sebelum mengisi formulir di bawah ini, sangat disarankan untuk membaca dan memahami informasi jalur pendaftaran yang dapat Anda temukan di [SINI](#)

**A. DATA UMUM** tidak boleh kosong

Gelombang Pendaftaran

NISN

**B. DATA PRIBADI** tidak boleh kosong

Nomor Induk Kependudukan

Nama Lengkap

Jenis Kelamin

Golongan Darah

Tempat Lahir

Tanggal Lahir

Agama

Kewarganegaraan

**C. DATA PENDIDIKAN TERKAHIR** tidak boleh kosong

Nama Ayah/Wali

Tanggal Lahir Ayah/Wali

Nama Ibu

Tanggal Lahir Ibu

Pekerjaan Orang Tua/Wali

ID Propinsi Asal

Alamat Asal

Telepon

Email

**D. PROGRAM STUDI PILIHAN** tidak boleh kosong

Pilihan I

Pilihan II

**E. PASSWORD UNTUK EDIT / CETAK DATA** tidak boleh kosong

Password

Dari manakah Anda mengetahui [Universitas Dian Nuswantoro](#) ?  
☐ Teman ☐ Keluarga ☐ Forum ☐ Google ☐ Media Cetak ☐ Baito ☐ Youtube ☐ Facebook

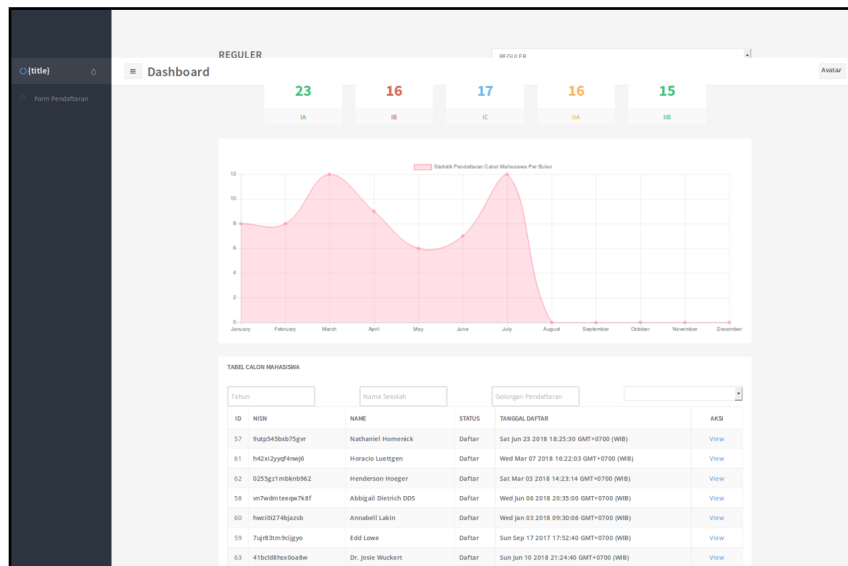
Pastikan data yang sudah anda isikan adalah benar sebelum melakukan penyimpanan formulir

**Simpan Formulir**

Gambar 5.18 Form pendataran mahasiswa baru Universitas Dian Nuswantoro

Halaman form pendaftaran mahasiswa baru Universitas Dian Nuswantoro ini menggunakan satu fungsi graphql yaitu createMerchant dimana fungsi ini digunakan untuk menyimpan data yang diinputkan oleh calon mahasiswa. Ketika data berhasil disimpan akan muncul pesan berhasil disimpan.

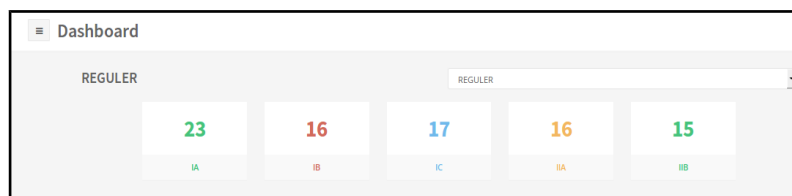
## 2. Halaman admin



Gambar 5.19 Halaman admin

Pada halaman admin dibagi menjadi tiga bagian yang pertama statistik jumlah pendaftar berdasarkan tipe pendaftarannya, statistik jumlah pendaftar berdasarkan bulan dan tahun pendaftaran, dan juga list data calon mahasiswa yang mendaftar.

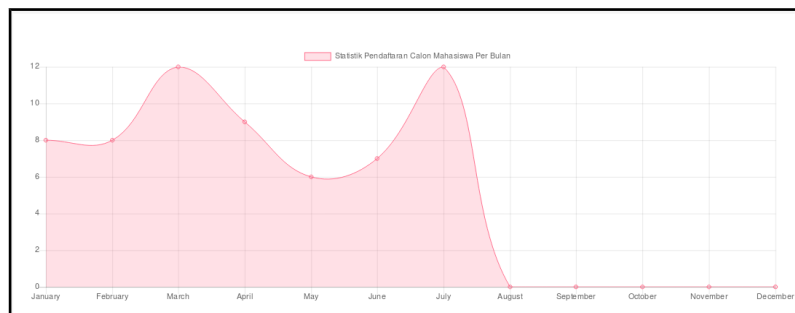
### a. Statistik jumlah pendaftar berdasarkan tipe pendaftaran



Gambar 5.20 Statistik jumlah pendaftar berdasarkan tipe pendaftaran

Pada bagian ini menggunakan fungsi `graphql sortMatriculant`, halaman ini akan menampilkan jumlah calon mahasiswa yang mendaftar berdasarkan tipe pendafrannya. Untuk mengganti tipe pendafrannya dapat mengganti tipenya pada menu *drop down* seperti pada gambar diatas.

- b. Statistik jumlah pendaftar berdasarkan bulan dan tahun sekarang



Gambar 5.21 Statistik jumlah pendaftar berdasarkan bulan dan tahun sekarang

Gambar merupakan statistik jumlah pendaftar berdasarkan bulan dan tahun sekarang, bagian ini menggunakan fungsi `graphql matriculantPerMonth`.



## c. List data pendaftar

| TABEL CALON MAHASISWA |                  |                       |        |   |                      |
|-----------------------|------------------|-----------------------|--------|---|----------------------|
| Tahun                 |                  | Nama Sekolah          |        | Golongan Pendaftaran                    |                      |
| ID                    | NISN             | NAME                  | STATUS | TANGGAL DAFTAR                          | AKSI                 |
| 57                    | 9utp545bsb75gyr  | Nathaniel Homenick    | Daftar | Sat Jun 23 2018 18:25:30 GMT+0700 (WIB) | <a href="#">View</a> |
| 61                    | h42xi2yyqf4nwj6  | Horacio Luetgen       | Daftar | Wed Mar 07 2018 16:22:03 GMT+0700 (WIB) | <a href="#">View</a> |
| 62                    | 0255gz1mbknb962  | Henderson Hoeger      | Daftar | Sat Mar 03 2018 14:23:14 GMT+0700 (WIB) | <a href="#">View</a> |
| 58                    | vn7wdmtteeqw7k8f | Abbigail Dietrich DDS | Daftar | Wed Jun 06 2018 20:35:00 GMT+0700 (WIB) | <a href="#">View</a> |
| 60                    | hwci0i274bjazsb  | Annabell Lakin        | Daftar | Wed Jan 03 2018 09:30:06 GMT+0700 (WIB) | <a href="#">View</a> |
| 59                    | 7ujr83tm9cjjgyo  | Edd Lowe              | Daftar | Sun Sep 17 2017 17:52:40 GMT+0700 (WIB) | <a href="#">View</a> |
| 63                    | 41bcd8hsx0oa8w   | Dr. Josie Wuckert     | Daftar | Sun Jun 10 2018 21:24:40 GMT+0700 (WIB) | <a href="#">View</a> |
| 64                    | 4vk1zdr84rx27es  | Modesto Bahringer     | Daftar | Sun Jan 28 2018 10:38:41 GMT+0700 (WIB) | <a href="#">View</a> |

Gambar 5.22 List data pendaftar

Bagian ini menampilkan seluruh data calon mahasiswa yang mendaftar. Pada bagian ini terdiri dari tiga bagian lain, bagian bagiannya antara lain pencarian calon mahasiswa dan list data calon mahasiswa. Bagian yang menampilkan daftar data calon mahasiswa menggunakan fungsi `graphql matriculantAll` dimana hanya *me-retrieve* lima atribut. Bagian atas dari tabel yang berisi data calon mahasiswa ialah bagian pencarian berdasarkan empat kategori, empat kategori itu ialah mencari calon mahasiswa berdasarkan tahun, asal sekolah, tipe pendaftaran dan status calon mahasiswa itu sendiri, bagian pencarian ini menggunakan fungsi `statMatriculant`, fungsi ini akan *me-retrieve* data yang sama persis seperti fungsi `matriculantAll retrieve`. Untuk menampilkan detail dari data calon mahasiswa.

## d. Detail data calon mahasiswa

| FORM ID : [FORM ID]      |  | Tanggal Lahir Ibu                       |   |
|--------------------------|--|---|---|
| Status                   | DAFTAR                                       | Mon Apr 30 2018 06:00:59 GMT+0700 (WIB) |   |
| Ganti Status             | Daftar                                       | Pekerjaan Orang Tua/Wali                | Customer Communications Administrator   |
| <b>A. DATA UMUM</b>      |  | Propinsi Asal                           | Bedfordshire                            |
| Golongan Pendaftaran     | IB   | Kota/Kabupaten Asal                     | Port Elnor                              |
| Jenis Pendaftaran        | regular                                      | Kecamatan Asal                          | Emmanuel Center                         |
| NISN                     | 9uip543bcb75gr                               | Alamat Asal                             | 39004 Damian Pines                      |
| <b>B. DATA PRIBADI</b>   |  | Kode Pos                                | 01806-7462                              |
| Nomor Induk Kependudukan | 3lnczmp8ho1kmqk                              | Telepon                                 | +629438993                              |
| Nama Lengkap             | Nathaniel Homernick                          | Email                                   | Bryce_Stromani80@gmail.com              |
| Jenis Kelamin            | FEMALE                                       | <b>C. DATA PENDIDIKAN TERAKHIR</b>      |   |
| Golongan Darah           | A  | Sekolah                                 | SMA N 1 SEMARANG                        |
| Tempat Lahir             | Cambridgeshire                               | Jurusan                                 | IPS                                     |
| Tanggal Lahir            | Sun Sep 17 2017 04:29:43 GMT+0700 (WIB)      | <b>D. PROGRAM STUDI PILIHAN</b>         |   |
| Agama                    | BUDHA  | Pilihan I                               | Sastra Inggris S1                       |
| Kewarganegaraan          | South Georgia and the South Sandwich Islands | Pilihan II                              | Teknik Elektro S1                       |
| <b>E. SURVEY</b>         |  | <b>F. SURVEY</b>                        |   |
| Nama Ayah/Wali           | Emiliano Turner                              | Sumber Informasi                        | YOUTUBE                                 |
| Tanggal Lahir Ayah/Wali  | Fri Jul 28 2017 13:18:01 GMT+0700 (WIB)      | Tanggal Daftar                          | Sat Jun 23 2018 18:25:30 GMT+0700 (WIB) |
| Nama Ibu                 | Lucious Purdy                                |   |   |

Gambar 5.23 Detail data calon mahasiswa

Bagian ini merupakan detail dari data calon mahasiswa sekaligus pada bagian ini dapat mengganti status calon mahasiswa. Untuk menampilkan detail data calon mahasiswa menggunakan fungsi `graphql matriculant` dimana memerlukan parameter `id`, sedangkan pada bagian ganti status calon mahasiswa menggunakan fungsi `graphql changeStatusMatriculant`.

## 5.2 Pengujian

### 5.2.1 White-box Testing

Untuk melakukan pengujian *white-box* penulis akan menggambar notasi sederhana untuk representasi aliran kontrol yang disebut grafik alir dan menentukan hasil kompleksitas siklomatik dari grafik alir tersebut. Pengujian ini dilakukan pada bagian fungsi yang paling penting, yaitu fungsi `statMatriculant`, `filterMonth`.

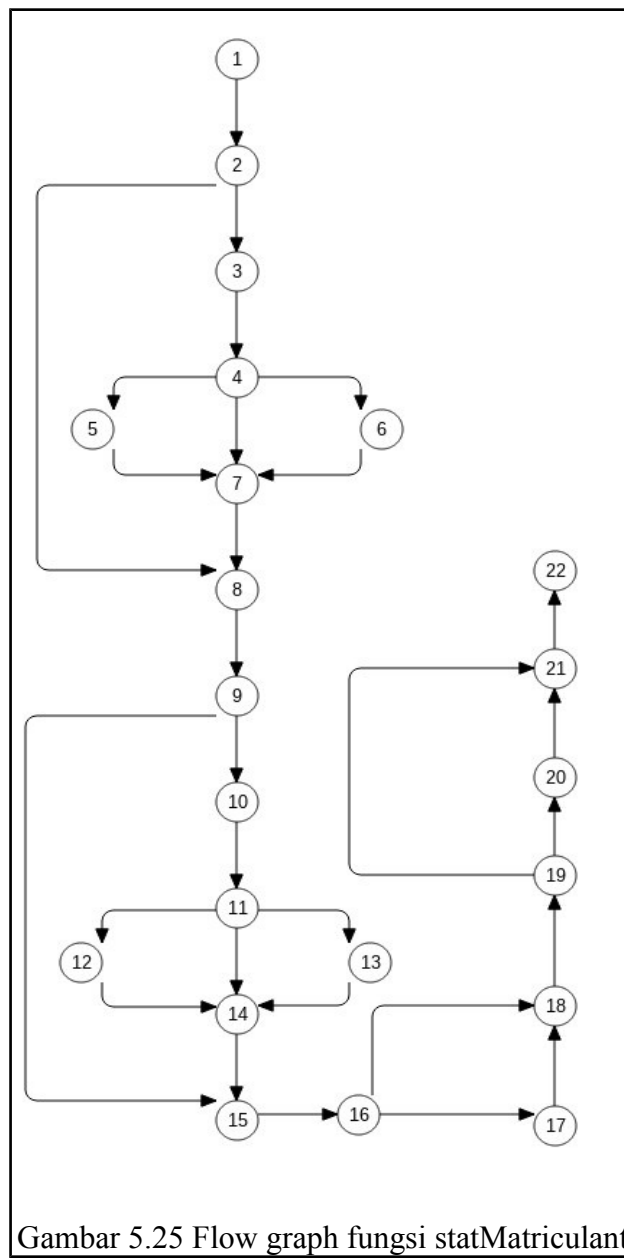
## 1. Fungsi statMatriculant

```

1 matriculantStatistic: async (_, {date, schoolName, regisGroup, status}) => {
2   let dateToInt = parseInt(date)
3   let Op = sequelize.Op
4   1 let findSchool = await models.LastEducation.findOne({
5     where: {schoolName: schoolName}})
6   let search = {}
7   2 if(regisGroup){
8     let findRegisGroup = await models.RegistrationGroup.findOne({
9       3 where: {group: regisGroup}})
10    4 if(findRegisGroup){
11      5 search.registrationGroupId = findRegisGroup.id
12    } else {
13      6 search.registrationGroupId = 0
14    }
15  }
16  8 }
17  9 if(schoolName){
18    let findSchool = await models.LastEducation.findOne({
19      10 where: {schoolName: schoolName}})
20    11 if(findSchool){
21      12 search.lastEducationId = findSchool.id
22    } else {
23      13 search.lastEducationId = 0
24    }
25  }
26  14 }
27  15 }
28  16 if(date){
29    let createdAt = {}
30    let dateTime = new Date()
31    let dateTimeNextMonth = new Date()
32    let year = parseInt(date)
33    let month = 0
34    17 dateTime.setFullYear(year)
35    dateTime.setMonth(0)
36    dateTime.setDate(1)
37    dateTimeNextMonth.setFullYear(year)
38    dateTimeNextMonth.setMonth(11)
39    dateTimeNextMonth.setDate(31)
40    createdAt = {
41      [Op.lt]: dateTimeNextMonth,
42      [Op.gt]: dateTime
43    }
44    search.createdAt = createdAt
45  }
46  18 }
47  19 if(status){
48    20 search.status = status
49  }
50  21 }
51  let findMatriculant = await models.Matriculant.findAll({
52    where: { $and: search },
53    include: [
54      {model: models.LastEducation},
55      {model: models.Origin},
56      {model: models.RegistrationGroup}
57    ]
58  })
59  return findMatriculant
60 },

```

Gambar 5.24 Potongan kode query statMatriculant



Jalur yang dihasilkan

P1:1-2-3-4-7-8-9-10-11-14-15-16-17-18-19-20-21-22

P2:1-2-8-9-15-16-18-19-21-22

P3:1-2-3-4-5-7-8-9-10-11-12-14-15-16-17-18-19-20-21-22

P4:1-2-3-4-6-7-8-9-10-11-13-14-15-16-17-18-19-20-21-22

P5:1-2-3-4-5-7-8-9-15-16-18-19-20-21-22

P6:1-2-3-4-6-7-8-9-15-16-18-19-20-21-22

P7:1-2-3-4-5-7-8-9-10-11-12-14-15-16-18-19-20-21-22

P8:1-2-3-4-6-7-8-9-10-11-12-14-15-16-18-19-20-21-22

P9:1-2-8-9-15-16-17-18-19-21-22

Cyclomatic Complexity

$$V(G) = E - N + 2$$

$$V(G) = 29 - 22 + 2$$

$$V(G) = 9$$

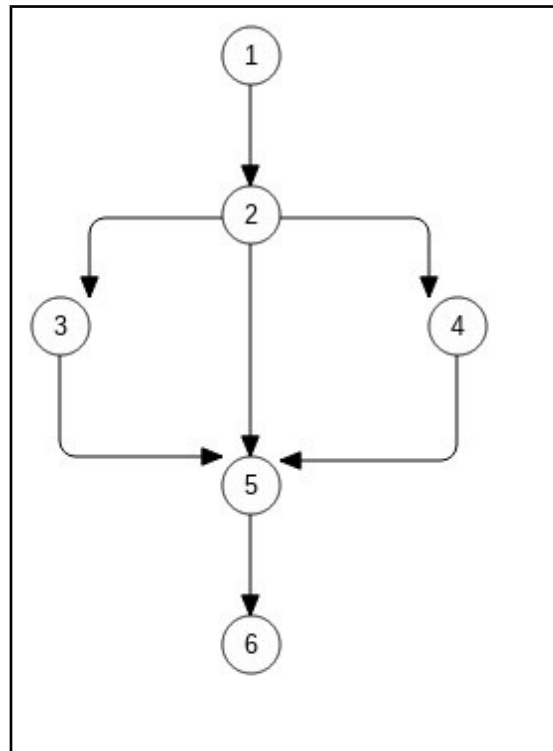
## 2. Fungsi filterMonth

```

1 let filterMonth= (matriculants,month)=>{
2   let dateTime= new Date()
3   1 dateTime.setMonth(month)
4   dateTime.setDate(1)
5   let dateTimeNextMonth= new Date()
6   2 if(month+1==11){
7     3 dateTimeNextMonth.setMonth(0)
8   }else{
9     4 dateTimeNextMonth.setMonth(month+1)
10  5 }
11  dateTimeNextMonth.setDate(1)
12  let filterMatriculant= matriculants.filter(matriculant=>{
13  6   return ((matriculant.createdAt > dateTime) && (matriculant.createdAt< dateTimeNextMonth))
14  })
15  return filterMatriculant.length
16 }

```

Gambar 5.26 Potongan kode fungsi filterMonth



Gambar 5.27 Flow graph fungsi  
filterMonth

Jalur yang dihasilkan

P1:1-2-5-6 P2:1-2-3-5-6 P3:1-2-4-5-6

Cyclomatic Complexity


$$V(G) = E - N + 2$$

$$V(G) = 7 - 6 + 2$$

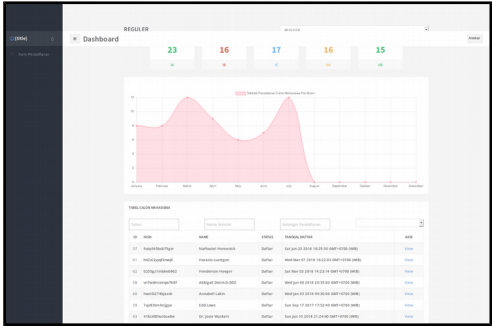
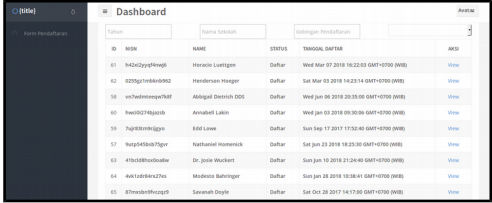
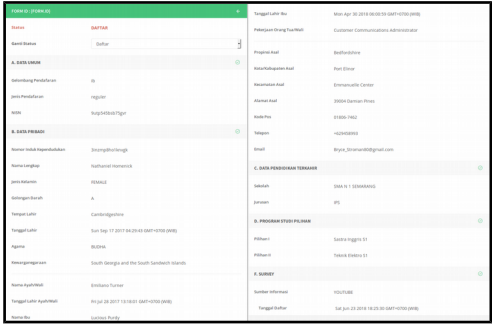
$$V(G) = 3$$

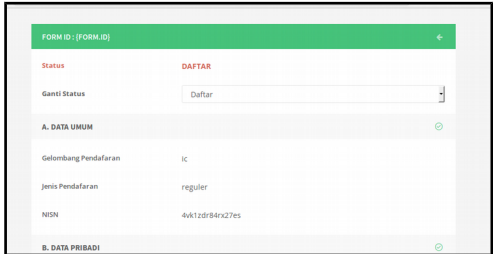
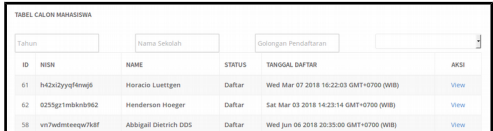
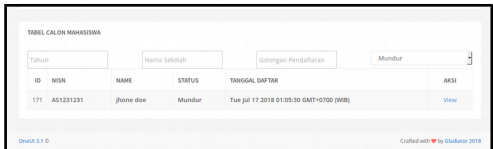
### 5.2.2 Black-box Testing

Black-Box Testing Aplikasi untuk sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro. Di bawah ini merupakan tabel hasil pengujian aplikasi menggunakan *Black-Box Testing*.

| No. | Modul            | Hasil yang diharapkan  | Hasil Pengujian   |
|-----|------------------|--|---|
| 1.  | Form pendaftaran | Menampilkan halaman form pendaftaran<br><br>ketika berhasil akan muncul tampilan seperti dibawah | Sesuai<br><br> |



|    |   |  |        |
|----|---|--|--------|
| 2. | Halaman Admin                           | <p>Menampilkan halaman admin yang berisi tentang informasi statistik dan daftar data calon mahasiswa</p>   | Sesuai |
| 3. | Menampilkan detail data calon mahasiswa | <p>Dibagian daftar data calon mahasiswa, ada tombol view digunakan untuk melihat detail data calon mahasiswa</p>  <p>ketika berhasil ditekan maka akan muncul seperti ini</p>  | Sesuai |

|    |  |  |        |
|----|--|--|--------|
| 4. | Mengganti status calon mahasiswa                     | <p>Untuk mengganti status, perlu mengganti lewat kolom ganti status status langsung tersimpan tanpa, halaman melakukan <i>reload</i>, seperti gambar dibawah</p>   | Sesuai |
| 5. | Menyaring data calon mahasiswa dengan empat kategory | <p>Untuk melakukan penyaringan, perlu mengisi empat inputan diatas tabel daftar calon mahasiswa</p>  <p>jika berhasil maka data yang akan tampil sesuai dengan yang sudah dikategorikan</p>  | Sesuai |

### 5.2.3 Performa GraphQL

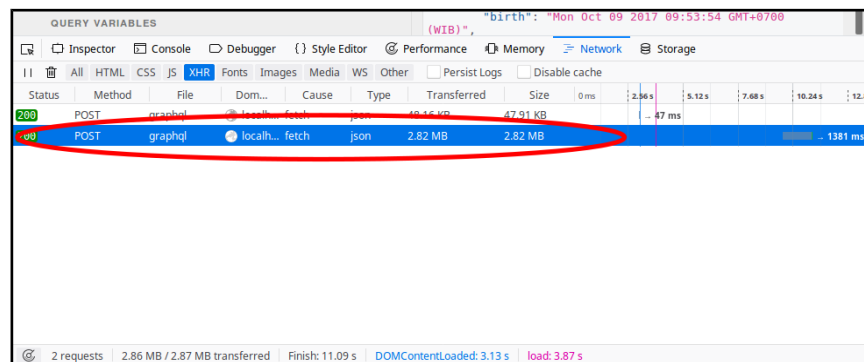
Dalam pembahasan kali ini peneliti akan mengambil salah satu *function* pada sistem yang telah dikembangkan untuk melihat ukuran data yang diambil dari *request* dan juga seberapa *bandwidth* yang digunakan. Penulis menggunakan *query matriculantAll* untuk melakukan pengujian ini. Dengan melakukan beberapa *request* menuju ke *query matriculantAll* dan menggunakan atribut *response* yang berbeda beda. Dengan melihat selisih *bandwidth* dan ukuran yang diperlukan dari setiap *request* yang digunakan. Ukuran ini sangat bergantung terhadap kebutuhan *client-side*. Dibawah ini merupakan contoh dari *query matriculantAll* dengan memberikan *response* seluruh data calon mahasiswa dengan atribut seperti dibawah.

```
query MatriculantAll {
  matriculantAll {
    id fullName NIK NISN
    gender bloodType birthPlace birth
    motherName motherBirth fatherName fatherName
    parentsJob email status religion
    citizenship phone password
    RegistrationGroup{
      type group createdAt updatedAt
    }
    LastEducation{
      schoolName major createdAt updatedAt
    }
    createdAt updatedAt
  }
}
```

Gambar 5.28 Request fungsi query matriculantAll

Untuk dapat mengetahui seberapa besar pengaruh pengurangan atribut pada setiap *response*. Penulis akan melakukan dua kali *request*, pertama melakukan *request* dengan mengirim seluruh atribut seperti pada gambar

diatas tanpa menggunakan paginasi dan *request* kedua mengirim lima atribut persis seperti apa yang ada pada *dashboard*, setelah melakukan dua kali *request* penulis akan perhitungan untuk mendapatkan selisih dari kedua *request* tersebut.



Gambar 5.29 Jumlah bandwidth dan ukuran respon dengan 21 atribut

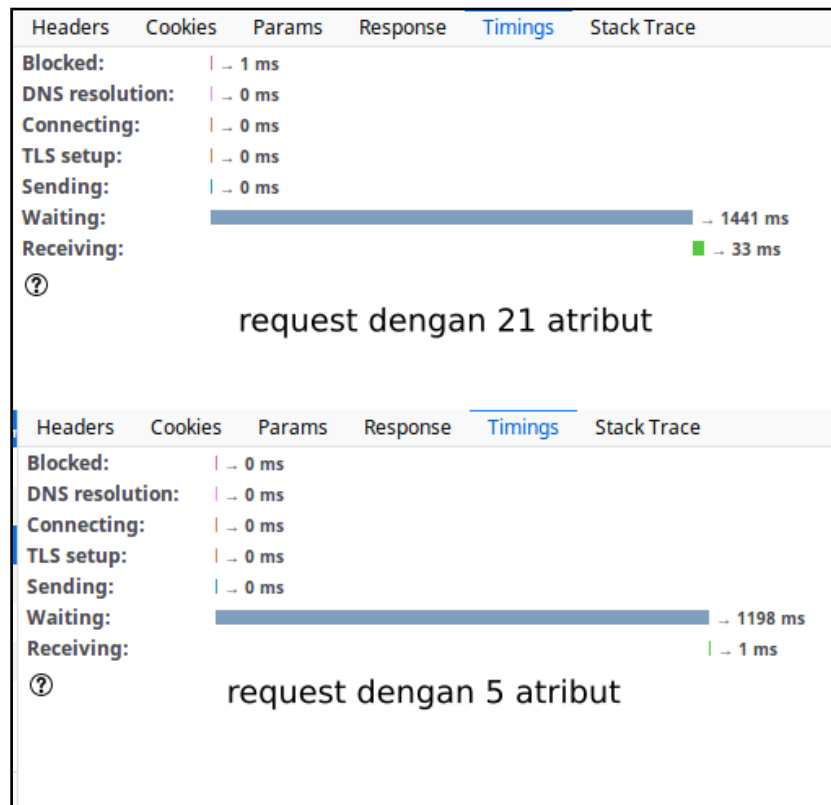
Gambar diatas dapat dilihat jumlah *bandwidth* dan ukuran *response* yang diterima sebesar 2,82 MB. Jumlah tersebut didapatkan dari melakukan request ke *query matriculantAll*, setelah itu penulis melakukan request ke query yang sama dengan mengirim lima atribut dan hasilnya bisa dilihat pada gambar dibawah. *Request* ini yang digunakan pada dashboard untuk menampilkan data calon mahasiswa. Ukuran *response* yang didapatkan dari melakukan *request* tersebut ialah 498,48 KB. Dengan hasil ini dapat disimpulkan bahwa dengan menggunakan graphql ukuran penggunaan *bandwidth* dan ukuran *response* berkurang hingga 82,32%. Besar dari hasil perhitungan ini sangat bergantung pada kebutuhan *client-side* yang mana memiliki kebutuhan berbeda beda.

| Status | Method  | File         | Dom...    | Cause | Type  | Transferred | Size      | 0 ms | 2.56 s | 5.12 s  | 7.68 s |
|--------|---------|--------------|-----------|-------|-------|-------------|-----------|------|--------|---------|--------|
| 204    | OPTIONS | graphql      | localh... | fetch | plain | 306 B       | 0 B       |      |        | 283 ms  |        |
| 204    | OPTIONS | graphql      | localh... | fetch | plain | 306 B       | 0 B       |      |        | 1 ms    |        |
| 204    | OPTIONS | graphql      | localh... | fetch | plain | 306 B       | 0 B       |      |        | 2 ms    |        |
| 200    | POST    | graphql      | localh... | fetch | json  | 353 B       | 108 B     |      |        | 2332 ms |        |
| 200    | POST    | graphql      | localh... | fetch | icon  | 424 B       | 179 B     |      |        | 2293 ms |        |
| 200    | POST    | graphql      | localh... | fetch | json  | 498.73 KB   | 498.48 KB |      |        | 3166 ms |        |
| 200    | GET     | info/1532... | localh... | xhr   | json  | 357 B       | 79 B      |      |        | 9 ms    |        |

7 requests 498.84 KB / 500.73 KB transferred Finish: 7.43 s DOMContentLoaded: 4.19 s load: 7.46 s

Gambar 5.30 Jumlah bandwidth dan ukuran pada query  
matriculantAll

Gambar dibawah merupakan selisih waktu yang didapatkan dari dua request ke fungsi yang sama tetapi memiliki jumlah atribut yang berbeda, dapat dilihat dari recevingnya selisih dari kedua *request* tersebut memiliki perbedaan yang sangat jauh dari 33ms menjadi 1ms. Selisih waktu ini sangatlah bergantung pada fungsi yang di-*request*, jumlah atribut, internet yang digunakan, dan masih banyak faktor yang mempengaruhi.



Tabel 5.1: Selishi latensi dari kedua request dengan atribut yang berbeda

## BAB VI

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Setelah melakukan penerapan GraphQL *web service* pada sistem penerimaan mahasiswa baru Universitas Dian Nuswantoro, kesimpulan dari penelitian ini adalah.

1. Dari pembahasan sebelumnya ketika penulis melakukan pengurangan atribut saat melakukan *request* ukuran yang *bandwidth* yang digunakan berkurang hingga 82,32%. Jadi dapat ditarik kesimpulan bahwa proses *request* dan *response* menjadi lebih cepat dan ringan karena dari sisi *client-side* dapat memilih sendiri atribut yang dibutuhkan.
2. *Vulnerability* HTTP *protocol* sudah dapat ditangani, karena GraphQL sudah tidak menggunakan url sebagai *endpoint request* dan *response*-nya, melainkan menggunakan fungsi. Hal ini membuat GraphQL menjadi lebih aman.

#### 6.2 Saran

Penelitian yang dilakukan tentunya tidak lepas dari kesalahan. Oleh karena itu penulis perlu memberikan saran untuk acuan peneliti selanjutnya. Berikut merupakan saran yang diberikan penulis:

1. Sistem *backend* API ini belum menggunakan keamanan dalam melakukan *request*-nya, seperti menggunakan JSON web token(JWT). JWT merupakan bentuk keamanan *web service* dimana untuk melakukan *request* diperlukan token akses, token ini dibuat

dari id user yang mengakses, tanggal kadaluarsa dan hak akses dari user tersebut. Hal ini menjadikan backend API tidak dapat diakses dengan mudah.

2. GraphQL menggunakan bentuk graph untuk *query*-nya, sedangkan basis data yang digunakan dalam penelitian ini ialah *relational database* dimana untuk melakukan *query* SQL. Diperlukan pembahasan lebih mendalam tentang *modelling* kompresi dari *query* SQL menjadi *query* graph.



## DAFTAR PUSTAKA

- [1]. “Penerimaan Mahasiswa Baru 2018 Universitas Dian Nuswantoro.” [Online]. Available: <https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>. [Accessed: 11-Dec-2017].
- [2]...... “Number of internet users worldwide from 2005 to 2017 (in millions),” *Statista*. [Online]. Available: <https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>. [Accessed: 11-Dec-2017].
- [3]. “Pertumbuhan Pengguna Internet, Indonesia Nomor 1 di Dunia,” Senin, Mei-2017. [Online]. Available: <https://databoks.katadata.co.id/datapublish/2017/05/22/pertumbuhan-pengguna-internet-indonesia-nomor-1-di-dunia>. [Accessed: 11-Dec-2017].
- [4]......Greg Sterling, “Report: Mobile Search Queries 29 Percent Of Total But Growth Modest.” [Online]. Available: <https://searchengineland.com/report-mobile-search-queries-29-percent-of-total-but-growth-modest-217501>. [Accessed: 11-Dec-2017].
- [5]...... “What Are RESTful Web Services?” [Online]. Available: <https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>. [Accessed: 11-Dec-2017].
- [6]David, “Rails 1.2: REST admiration, HTTP lovefest, and UTF-8 celebrations.” [Online]. Available: <http://weblog.rubyonrails.org/2007/1/19/rails-1-2-rest-admiration-http-lovefest-and-utf-8-celebrations/>. [Accessed: 11-Dec-2017].
- [7]......Simon E Spero, “Analysis of HTTP Performance problems.” [Online]. Available: <https://www.w3.org/Protocols/HTTP-NG/http-prob.html>.
- [8]...... “Optimizing for Mobile Networks.” [Online]. Available: <https://hpbn.co/optimizing-for-mobile-networks/#anticipate-network-latency-overhead>. [Accessed: 11-Dec-2017].

- [9] “Di Asia-Pasifik, Kecepatan Internet Indonesia Jauh Tertinggal,” 27-Mar-2017. [Online]. Available: <https://databoks.katadata.co.id/datapublish/2017/03/27/di-asia-pasifik-kecepatan-internet-indonesia-jauh-tertinggal>. [Accessed: 11-Dec-2017].
- [10].....O. Hartig and J. Pérez, “An Initial Analysis of Facebook’s GraphQL Language,” in *AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017.*, 2017, vol. 1912.
- [11].....E. Stenlund and K. Gustavsson, “Efficient data communication between a webclient and a cloud environment,” 2016.
- [12]...F. Nogatz and D. Seipel, “Implementing GraphQL as a Query Language for Deductive Databases in SWI-Prolog Using DCGs, Quasi Quotations, and Dicts,” *ArXiv Prepr. ArXiv170100626*, 2017.
- [13]. “EHRI Mission Statement.” [Online]. Available: <https://ehri-project.eu/about-ehri>. [Accessed: 27-Dec-2017].
- [14]...Mike Bryant, “GraphQL for Archival Metadata: An Overview of the EHRI GraphQL API,” presented at the Big Data, Boston, 2017.
- [15].....“Interoperability.” [Online]. Available: <https://en.wikipedia.org/wiki/Interoperability>. [Accessed: 27-Dec-2017].
- [16].....“Client–server model.” [Online]. Available: [https://techterms.com/definition/client-server\\_model](https://techterms.com/definition/client-server_model). [Accessed: 26-Dec-2017].
- [17].....“Apa itu World Wide Web ?” [Online]. Available: [http://faculty.petra.ac.id/dwikris/docs/desgrafisweb/www/4-apaitu\\_www.html](http://faculty.petra.ac.id/dwikris/docs/desgrafisweb/www/4-apaitu_www.html). [Accessed: 26-Dec-2017].
- [18].....“HTML.” [Online]. Available: <https://developer.mozilla.org/en-US/docs/Learn/HTML>. [Accessed: 26-Dec-2017].
- [19].....“HTTP.” [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP>. [Accessed: 26-Dec-2017].

- [20].....“What Are Web Services?” [Online]. Available: <https://docs.oracle.com/javaee/6/tutorial/doc/gijvh.html>. [Accessed: 26-Dec-2017].
- [21].....“Do you know what a REST API is?” [Online]. Available: <https://www.sitepoint.com/developers-rest-api/>. [Accessed: 26-Dec-2017].
- [22]..“Pengertian API (Application Programming Interface).” [Online]. Available: <http://developer.erabelajar.com/api-application-programming-interface/>. [Accessed: 26-Dec-2017].
- [23].....“GraphQL.” [Online]. Available: <http://graphql.org/>. [Accessed: 26-Dec-2017].
- [24]...“Introducing JSON.” [Online]. Available: <https://www.json.org/>. [Accessed: 26-Dec-2017].
- [25].....“Deductive Databases.” [Online]. Available: <http://www3.cs.stonybrook.edu/~warren/xsbbook/node12.html>. [Accessed: 26-Dec-2017].
- [26].....“Relational database.” [Online]. Available: [https://en.wikipedia.org/wiki/Relational\\_database](https://en.wikipedia.org/wiki/Relational_database). [Accessed: 26-Dec-2017].
- [27].....“NoSQL.” [Online]. Available: <https://en.wikipedia.org/wiki/NoSQL>. [Accessed: 26-Dec-2017].
- [28].....“Node.js programs versus PHP.” [Online]. Available: <https://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=node&lang2=php>. [Accessed: 14-Jan-2018].
- [29].....S. Nidhra and J. Dondeti, “Blackbox and whitebox testing techniques-a literature review,” *Int. J. Embed. Syst. Appl. IJESA*, vol. 2, no. 2, pp. 29–50, 2012.
- [30].....M. E. Khan, “1 Different Approaches to White Box Testing Technique for Finding Errors,” 2011.