KERJA PRAKTIK – KI141330

Analisis Sistim Human Resources (HR) dan Customer Relationship Management (CRM) PT. Gamatechno Indonesia

PT Gamatechno Indonesia

Oleh:
ALIFA RIDHO MUSTHAFA 5112100045
ALIEF YOGA PRIYANTO 5112100211

Pembimbing Jurusan
Dwi Sunaryono, S.Kom.,M.Kom.
Pembimbing Lapangan
A. Toto Priyono

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – KI141330

Analisis Sistim Human Resources (HR) dan Customer Relationship Management (CRM) PT. Gamatechno Indonesia

PT Gamatechno Indonesia

Oleh:

ALIFA RIDHO MUSTHAFA	5112100045
ALIEF YOGA PRIYANTO	5112100211

Pembimbing Jurusan
Dwi Sunaryono, S.Kom.,M.Kom.
Pembimbing Lapangan
A. Toto Priyono

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

Analisis Sistim Human Resources (HR) dan Customer Relationship Management (CRM) PT. Gamatechno Indonesia

Oleh:

ALIFA RIDHO MUSTHAFA	5112100045
ALIEF YOGA PRIYANTO	5112100211

Disetujui oleh Pembimbing Kerja Praktik:

1. Dwi Sunaryono, S.Kom.,M.Kom.
   NIP. 19720528 199702 1 001
   ..... (Pembimbing Jurusan)

2. A. Toto Priyono
   NIP. 2009.040178.0212
   ..... (Pembimbing Lapangan)

GAMATECHNO
Enhancing Your Business

SURABAYA
JULI, 2015

[Halaman ini sengaja dikosongkan]

Analisis Sistim Human Resources (HR) dan Customer Relationship Managemen (CRM)

Nama Mahasiswa : Alifa Ridho Musthafa
NRP : 5112100045
Nama Mahasiswa : Alief Yoga Priyanto
NRP : 5112100211
Jurusan : Teknik Informatika FTIf-ITS
Pembimbing Jurusan : Dwi Sunaryono, S.Kom.,M.Kom.
Dosen Pembimbing II : A. Toto Priyono

ABSTRAK

PT Gamatechno Indonesia merupakan perusahaan yang bergerak dibidang perusahaan yang memiliki fokus pada pengembangan produk dan solusi teknologi informasi untuk segmen perguruan tinggi, lembaga pemerintah, perusahaan penyedia jasa transportasi dan logistik, serta industri lifestyle. PT Gamatechno memliki banyak produk, diantaranya adalah Sistem gtEnterprise Human Resources (gtHR) dan Sistem Simple Customer Relationship Management (SimpleCRM).

GtEnterprise Human Resources atau disingkat gtHR adalah aplikasi web yang memberikan kemudahan dalam pengelolaan data karyawan, pencatatan proses mutasi, penilaian kerja, penyederhanaan pengelolaan gaji dan penghitungan PPh (Pajak Penghasilan) serta melakukan rekapitulasi kehadiran karyawan. Software yang dibangun dengan Gamatechnno Framework (GTFW) ini memiliki fitur yang kompleks dan lengkap. Namun, masih memiliki kekurangan pada tampilan struktur organisasi yang kurang menarik. Sedangkan SimpleCRM atau simple adakah aplikasi yang memudahkan pengelolaan customer, mitra bisnis dan prospek perusahaan, memberikan pertimbangan kepada eksekutif perusahaan terkait proyek yang ada dalam perusahaan. Aplikasi ini mendukung

banyak platform berkat bantuan web service. Sayangnya, web service ini berlum tersedia.

Untuk meningkatkan tampilan struktur organisasi pada aplikasi gtHR, digunakan organization chart dengan bantuan JQuery/CSS. Untuk membangun web service SimpleCRM, dipilihlah framework slimeframework. Framework ini sangat mudah dipelajari dan terdokumentasikan dengan baik serta sangat mudah untuk dikembangkan.

Sekarang aplikasi gtHR mampu menampilkan struktur organisasi dengan organization chart. Aplikasi SimpleCRM juga telah didukung web service yang membuat aplikasi ini mampu berjalan diberbagai platform, khususnya mobile.

Kata kunci: [Sistem gtEnterprise Human Resources], [Sistem Simple Customer Relationship Management], [Gamatechnno Framework], [Slimframework], [web service], [organization chart]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat dan karunianya, sehingga penulis dapat menyelesaikan Laporan Pelaksanaan Kerja Praktek ini.

Kerja Praktek ini merupakan salah satu matakuliah yang wajib ditempuh di Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS). Laporan Kerja Praktek ini disusun sebagai pelengkap kerja praktek yang telah dilaksanakan lebih kurang 1 bulan di PT Gamatechno Indonesia.

Dengan selesainya laporan kerja praktek ini tidak terlepas dari bantuan banyak pihak yang telah memberikan masukan-masukan kepada penulis. Untuk itu penulis mengucapkan banyak terimakasih kepada :

1. Bapak Dwi Sunaryono selaku Dosen Pembimbing
2. Bapak A. Toto Priyono selaku Pembimbing lapangan
3. Karyawan PT Gamatechno Indonesia

Penulis menyadari bahwa masih banyak kekurangan dari laporan ini, baik dari materi maupun teknik penyajiannya, mengingat kurangnya pengetahuan dan pengalaman penulis. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan.

Terimakasih.

Surabaya, Juli 2015

Alifa Ridho Musthafa dan Alief Yoga Priyanto

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1. Latar Belakang

PT. Gamatechno merupakan perusahaan yang bergerak dibidang perusahaan yang memiliki fokus pada pengembangan produk dan solusi teknologi informasi untuk segmen perguruan tinggi, lembaga pemerintah, perusahaan penyedia jasa transportasi dan logistik, serta industri lifestyle. PT. Gamatechno membuat perangkat lunak bersifat generik maupun pesanan. Banyak sekali perangkat lunak yang sudah tercipta maupun yang sedang dalam pengerjaan. Dua diantaranya adalah sistim gtHR(gamatechno *Human Resourses)* dan SimpleCRM(*Simple Customer Relationship Management).* Kedua perangkat lunak tersebut merupakan produk yang bersifat generik atau umum.

Aplikasi gtHR merupakan aplikasi yang menyediakan fitur yang memberikan kemudahan dalam pengelolaan data karyawan, pencatatan proses mutasi, penilaian kerja, penyederhanaan pengelolaan gaji dan penghitungan PPh (Pajak Penghasilan) serta melakukan rekapitulasi kehadiran karyawan . Dikembangkan dengan gamatechno framework(gtfw), yakni framework buatan PT. Gamatechno. Perangkat lunak ini memiliki banyak sekali fitur dan sudah berada pada tahap akhir pengembangan, namun masih ada beberapa kekurangan yang bersifat minor. Aplikasi SimpleCRM adalah aplikasi yang memudahkan pengelolaan *customer*, mitra bisnis dan prospek perusahaan, memberikan pertimbangan kepada eksekutif perusahaan terkait proyek yang ada dalam perusahaan. Aplikasi ini mampu mendukung banyak *platform* dan *device* dengan bantuan web service. Sayangnya web service ini belum tersedia.

Dengan didasari hal – hal diatas, maka kerja praktik bertujuan untuk menganalisa, memperbaiki, dan menambah fitur dari aplikasi gtHR serta membuat *webservice* untuk aplikasi simpleCRM.

1.2. Tujuan

Adapun tujuan dari kerja praktik ini adalah:

1. Menampilkan struktur organisasi dalam bentuk *organizational chart.*
2. Mendapatkan akses *database* sistem aplikasi simpleCRM dari berbagai *platform*.

1.3. Manfaat

1. Mengetahui struktur GTFW (*Gamatechno Framework)* dan dapat mengimplementasikan dalam rekayasa perangkat lunak.
2. Memudahkan pelacakan posisi pegawai dalam suatu instansi dengan menggunakan *organizational chart.*
3. Memudahkan akses database sistem untuk aplikasi berbasis *desktop* maupun *mobile*.

1.4. Rumusan Permasalahan

1. Bagaimana cara menampilkan struktur organisasi dalam aplikasi gtHR?
2. Bagaimana cara menampilkan daftar pegawai pada struktur organisasi dalam aplikasi gtHR?
3. Bagaimana agar *database* dapat dengan mudah diakses oleh aplikasi dari *platform mobile* SimpleCRM ?

1.5. Lokasi dan Waktu Kerja Praktik

Tempat Pelaksanaan

Lokasi	: PT Gamatechno Indonesia
Alamat	: Jalan Cik Di Tiro no. 34, Yogyakarta 55223, Indonesia

Waktu pelaksanaan

Durasi	: 10 Juni – 10 Juli 2015
Jam mulai	: 08.00 WIB
Jam selesai	: 16.30 WIB (hari biasa) / 15.30 WIB (bulan Ramadhan)

Kehadiran : harus hadir setiap hari dari senin – jumat.

1.6. Metodologi Kerja Praktik

Berisi tahapan pengerjaan kerja praktik secara rinci, apa saja yang dilakukan untuk setiap prosesnya (perumusan masalah, studi literatur, analisis dan perancangan sistem, implementasi sistem, pengujian dan evaluasi, kesimpulan dan saran)

1. **Perumusan Masalah**

   Perumusan masalah aplikasi gtHR dan simpleCRM dilakukan dengan cara diskusi dengan tim dan pembimbing lapangan. Diskusi dilakukan untuk membahas kekurangan-kekurangan dan celah yang ada pada kedua aplikasi serta mengidentifikasi fungsi-fungsi yang diperlukan untuk memenuhi kebutuhan pengguna.

2. **Studi Literatur**

   Studi literatur dilakukan untuk mempelajari struktur dan implementasi GTFW secara bertahap. Studi juga dilakukan untuk mengetahui cara pengkodean atau pembuatan API *web service* dengan metode RESTful. Selain itu, dipelajari juga pengaplikasian kerangka kerja Slim Framework untuk menerapkan RESTful API *web service* dengan bahasa pemrograman php.

3. **Analisis dan Perancangan Sistem**

   Di tahap ini, pada aplikasi gtHR dilakukan penelurusuran sumber permasalahan dari *bug-bug* yang telah ditemukan sebelumnya bersama tim. Pencarian dilakukan di setiap modul yang ditemukan *bug* mulai dari lapisan *template, response,* hinnga *business.* Sumber permasalahan kemudian disesuaikan dengan kebutuhan fungsional maupun non-fungsional sistem. Apabila tidak sesuai, dicari solusi untuk setiap permasalahan yang ditemukan. Selain itu dilakukan juga penggalian kebutuhan untuk fungsi *organizational chart* untuk kemudian

dikonsultasikan kepada pihak Gamatechno. Tahap ini menghasilkan kebutuhan sistem untuk fungsi *organizational chart.* Selain itu juga menghasilkan daftar permasalahan dan solusinya.

Pada aplikasi simpleCRM, dilakukan penggalian kebutuhan sistem. Data-data yang dibutuhkan oleh sistem untuk pertukaran data didefinisikan kemudian dianalisis kebutuhan fungsinya apakah menggunakan method GET, POST, PUT atau DELETE. Pada tahap ini dihasilkan daftar kebutuhan sistem API *web service* untuk aplikasi simpleCRM.

4. Implementasi Sistem

Solusi-solusi dari hasil analisis diterapkan pada sistem gtHR. Solusi diimplementasikan dengan menerapkan *framework* GTFW untuk masing-masing *bug* yang ditemukan hingga kebutuhannya tercapai. Sedangkan *organization chart* ditambahkan dengan *jquery* dan dikonfigurasikan ke sistem.

SimpleCRM diimplementasikan dengan menggunakan *framework* Slim. Metode yang digunakan adalah metode RESTful.

5. Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian pada kebutuhan fungsional maupun non-fungsional sistem. Masing-masing *bug* yang telah diperbaiki diuji sesuai fungsi masing-masing apakah masih ada kekuarangan atau tidak. *Organization chart* diuji apakah sudah dapat menampilkan bagan organisasi dari database dengan baik dan sesuai dengan kebutuhan sistem atau belum.

Sedangkan pada simpleCRM pengujian dilakukan dengan aplikasi *plugin* dari Google Chrome yaitu Postman. Postman digunakan untuk menguji valid atau tidakkah kode yang dibuat. Selain itu Postman juga digunakan untuk mengecek keluaran yang dihasilkan apakah sudah sesuai dengan kebutuhan ataukah belum. Selanjutnya pengujian ke aplikasi dilakukan oleh pihak Gamatechno.

6. Kesimpulan dan Saran

Dari hasil pengujian dan evaluasi, dapat ditarik kesimpulan-kesimpulan yang mampu menjawab rumusan masalah yang ditemukan sebelumnya. Selain itu, terdapat saran-saran untuk perbaikan penelitian kerja praktik, metode yang digunakan, atau kepada perusahaan tempat kerja praktik dilaksanakan.

1.7. Sistematika Laporan

BAB I: PENDAHULUAN

Berisi pendahuluan yang terdiri dari latar belakang, permasalahan, tujuan dan manfaat proyek, batasan masalah, waktu pelaksanaan, dan sistematika penulisan yang digunakan dalam kerja praktik.

BAB II: PROFIL PERUSAHAAN

Bab ini berisi penjelasan mengenai sejarah, deksripsi dan visi mauapun misi dari perusahaan.

BAB III: TINJAUAN PUSTAKA

Bab ini berisi dasar teori dari metode/teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

BAB IV: ANALISIS DAN PERANCANGAN

Pada bab ini dijelaskan mengenai desain antar muka aplikasi.

BAB V: IMPLEMENTASI SYSTEM

Bab ini menjelaskan tentang kode implementasi yang telah dibuat.

BAB VI: UJI COBA DAN EVALUASI

Pada bab ini menjelaskan tentang proses ujicoba fungsi-fungsi perangkat lunak dan hasil evaluasinya.

BAB VII: KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari proses pelaksanaan tugas praktik.

BAB II
PROFIL PERUSAHAAN

2.1. Sejarah Perusahaan

Sebagai perusahaan yang bergerak di bidang penyedia solusi teknologi informasi, PT Gamatechno Indonesia (Gamatechno) resmi berdiri pada tanggal 4 Januari 2005 dan berkantor pusat di Yogyakarta. Guna meningkatkan layanan kepada lebih dari 240 klien di seluruh Indonesia yang tersebar dari Banda Aceh hingga Papua, pada tahun 2013 Gamatechno membuka kantor cabang di Jakarta.

Seiring dengan perkembangan perusahaan, saat ini Gamatechno memiliki fokus pada pengembangan produk dan solusi teknologi informasi untuk segmen perguruan tinggi, lembaga pemerintah, perusahaan penyedia jasa transportasi dan logistik, serta industri lifestyle. Layanan yang berfokus pada 4 segmen utama tersebut selanjutnya didefinisikan sebagai *gtSmartCity Solution*, yaitu solusi berbasis sistem dan teknologi informasi guna mewujudkan sebuah kota cerdas dengan ciri *less paper*, *less time*, *less cash* dan *less complexity* untuk meningkatkan tatanan hidup masyarakat.

Untuk segmen perguruan tinggi, produk unggulan Gamatechno adalah *gtCampus Suite* yaitu sistem informasi terintegrasi untuk perguruan tinggi yang terdiri atas berbagai software modular yang dirancang sesuai dengan proses bisnis perguruan tinggi mulai dari pengelolaan penerimaan calon mahasiswa, pengelolaan perkuliahan mahasiswa hingga lulus, pengelolaan aset kampus yang meliputi aset sumber daya manusia, keuangan dan aset barang, perpustakaan, penelitian dan beasiswa hingga *dashboard system* untuk pimpinan kampus.

Untuk segmen lembaga pemerintah, Gamatechno memiliki beberapa produk unggulan, diantaranya adalah gtPerizinan (sistem pengelolaan pelayanan perizinan terpadu), gtAspirasi (sistem pengelolaan aspirasi masyarakat), serta aplikasi gtGroupware (sistem kolaborasi dan arsip perkantoran). Selain produk-produk tersebut, Gamatechno juga melayani pengembangan portal website lembaga

dengan konsep *citizen centric*, serta pengembangan berbagai aplikasi berbasis web lainnya sesuai dengan kebutuhan lembaga.

Untuk segmen transportasi dan logistik, Gamatechno mengembangkan beberapa produk unggulan bagi perusahaan atau organisasi yang bergerak dibidang layanan transportasi dan logistik, yaitu gtFleets (sistem informasi pengelolaan armada), gtSmartTicket System (sistem tiket elektronik berbasis *smartcard*), serta aplikasi mTransport (aplikasi mobile untuk informasi dan layanan transportasi publik).

Pada segmen *lifestyle*, Gamatechno mengembangkan produk-produk aplikasi back-end dan front-end untuk beberapa sub industri diantaranya taman hiburan dan wisata, pusat belanja dan entertainment, microfinance, dan industri kesehatan. Beberapa portofolio produk untuk segmen lifestyle ini antara lain eoviz.com (small & medium enterprises resource planning system on cloud), mEvent (aplikasi mobile informasi event), serta mCatalog (aplikasi *mobile* informasi katalog produk).

Selain pengembangan produk aplikasi berbasis web, *mobile*, smartcard dan beberapa teknologi terkini lainnya yang dikemas dalam *gtSmartCity Solution*, Gamatechno juga menyediakan jasa konsultasi IT, audit IT, *training IT*, serta layanan *maintenance* sistem dan agregasi konten digital.

Sebagai perusahaan yang berbasis perguruan tinggi, Gamatechno memiliki keunggulan kompetitif yang tidak dimiliki perusahaan lain, yaitu sumber daya dan aset riset yang dimiliki Universitas Gadjah Mada sebagai dasar pengembangan dan inovasi produk serta layanan Gamatechno agar tercipta solusi yang tepat guna bagi masyarakat. Dan untuk melengkapi layanan total kepada pelanggan dan mitra, saat ini Gamatechno telah memiliki anak perusahaan yaitu PT Aino Indonesia yang bergerak dibidang teknologi *smartcard*, RFId, dan Mobile NFC.

2.2. Visi dan Misi Perusahaan

2.2.1. Visi

Menjadi perusahaan penyedia TI terbaik di Indonesia

2.2.2. Misi

Dalam rangka pencapaian visi dan tujuan perusahaan, maka Gamatechno menjabarkan misi-misi perusahaan sebagai berikut :

1. Mengakomodasi sumber daya, potensi, dan kebutuhan UGM
2. Mengembangkan solusi TIK yang berkesinambungan
3. Membangun jejaring kerjasama untuk menumbuhkan industri digital di Indonesia
4. Memberikan benefit dan value bagi mitra, pelanggan, dan seluruh *stakeholder*

2.3. Struktur Organisasi

Tabel 1 Struktur Organisasi

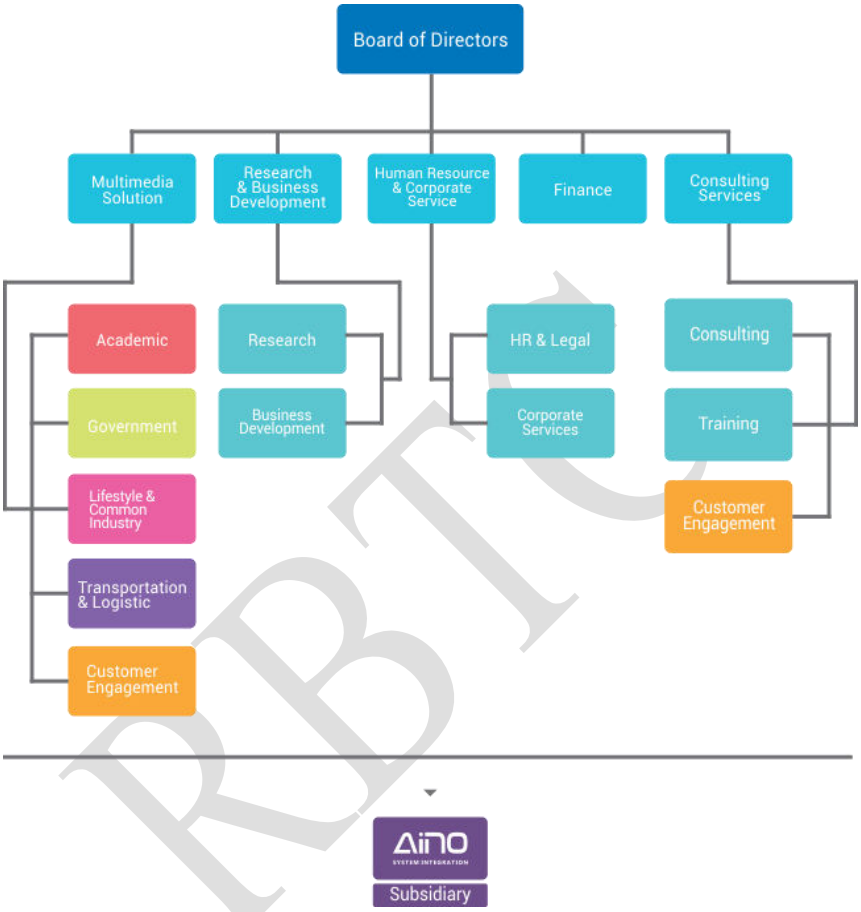
Komisaris Utama	:	DR. Didi Achjari, S.E., M.Com., Akt
Komisaris	:	Widyawan, S.T., M.Sc., Ph.D.
Komisaris	:	M. Afrizal Hernandar, S.T., MBA.
President Director	:	Muhammad Aditya A N
Director	:	Adityo Hidayat St. Majo Kayo, CISA

Research and Business Development General Manager	:	Novan Hartadi
Multimedia General Manager	:	Nanang Ruswianto
Consulting Services General Manager	:	Nugroho Setio Wibowo
Finance General Manager	:	Reni Nurika Andayani
Human Resource and Corporate Services General Manager	:	R. Sumarwan Ismunu
Business Development Manager	:	Triasmono
Academic Segment Manager	:	Awaludin Zakaria
Government Segment Manager	:	Taufik Suryawan Edyna
Lifestyle Segment Manager	:	A. Toto Priyono
Transportation Segment Manager	:	Alvonsius Albert Naipupu
Customer Engagement Manager	:	Yenni Eka Susanti
HR and Legal Manager	:	Andri Kushendarto
Branch Manager Jakarta	:	I.G.P. Rahman Desyanta

2.4. Divisi *Lifestyle and Common Industry*

Divisi *lifestyle and Common Industry* Gamatechno mengembangkan aplikasi di bidang *tourism* & taman hiburan, *shopping* & *entertainment*, microbanking dan industri estetika untuk memberikan pengalaman yang lebih menyenangkan bagi konsumennya. Contoh produk divisi *lifestyle* diantaranya adalah coviz.com, mEvent serta mCatalog.

Divisi ini dibawahi oleh divisi *multimedia solution* dan *board of directors,* seperti yang terlihat pada Gambar 2.1.



Gambar 2.1 Struktur Organisasi PT. Gamatechno

BAB III
TINJAUAN PUSTAKA

Pada bab ini, penulis menjelaskan tentang beberapa tinjauan pustaka yang menjadi dasar penulis dalam melakukan kerja praktek.

3.1. Sistem GtEnterprise Human Resources (gtHR)

gtHR adalah *software* yang memberikan kemudahan dalam pengelolaan data karyawan, pencatatan proses mutasi, penilaian kerja, penyederhanaan pengelolaan gaji dan penghitungan PPh (Pajak Penghasilan) serta melakukan rekapitulasi kehadiran karyawan. gtEnterprise HR dibangun dengan tujuan agar pengelola perusahaan mengetahui kondisi karyawan dan pada akhirnya dapat mengambil keputusan yang tepat terkait dengan strategi SDM yang akan di terapkan pada perusahaan tersebut.

Software gtHR dirancang agar dapat mengelola *database* presensi karyawan, baik yang dilakukan dengan sistem *barcode*, *fingerprint* dan *smartcard*, dan kemudian mengintegrasikannya dengan sistem *payroll* karyawan. Kerumitan dalam penghitungan gaji dan mengetahui kapan karyawan habis kontrak, dapat diselesaikan dengan aplikasi ini.

3.2. Sistem Simple Customer Relationship Management (SimpleCRM)

Mempermudah pengelolaan customer, mitra bisnis dan prospek perusahaan, memberikan pertimbangan kepada eksekutif perusahaan terkait proyek yang ada dalam perusahaan.

3.3. Gamatechno FrameWork

Gamatechno FrameWork atau yang lebih familiar disebut GTFW, adalah framework PHP yang dikembangkan oleh PT Gamatechno Indonesia. Gamatechno menggunakan GTFW dalam mengembangkan berbagai aplikasi berbasis web yang ditujukan untuk klien perguruan tinggi, pemerintahan maupun korporasi. Sejak dikembangkan pada tahun 2006 hingga saat ini, GTFW telah

mencapai versi 3 dengan penambahan dan perbaikan fitur pada versi sebelumnya.

3.4. Slim Framework

Slim adalah kerangka kerja PHP mikro yang membantu dalam membuat aplikasi web sederhana namun sangat membantu dan API.

3.5. Breadcrumb

Breadcrumb Navigation adalah sebuah menu navigasi/penunjuk arah halaman yang sedang dibuka berupa link horisontal berurut yang diawali dengan menu Home/Beranda >> Label >> Judul Halaman Yang Sedang Dibuka.

Mengenai fungsinya, navigasi breadcrumb ini memiliki fungsi yaitu untuk mempermudah pengunjung melacak lokasi dalam dokumen dan kembali ke awal halaman/home-page dengan mudah.

Menurut Jacob Nielsen, ditinjau dari sisi kegunaanya breadcrumb navigation berfungsi untuk menunjukkan lokasi/path dari halaman web/blog yang sedang dikunjungi, relatif terhadap struktur di atasnya serta menyediakan fitur *one click access* ke level halaman web/blog di atasnya sekaligus menghindari pengunjung tersesat karena terlalu dalam menjelajah web/blog kita.

3.6. RESTful API Web Service

REST adalah salah satu jenis web service yang menerapkan konsep perpindahan antar state. State disini dapat digambarkan seperti jika browser meminta suatu halaman web, maka serverakan mengirimkan state halaman web yang sekarang ke browser. Bernavigasi melalui link-link yang disediakan sama halnya dengan mengganti state dari halaman web. Begitu pula REST bekerja, dengan bernavigasi melalui link-link HTTP untuk melakukan aktivitas tertentu, seakan-akan terjadi perpindahan state satu sama lain. Perintah HTTP yang bisa digunakan adalah fungsi GET, POST, PUT atau DELETE. Balasan yang dikirimkan adalah dalam bentuk XML sederhana tanpa ada protokol pemaketan data, sehingga

informasi yang diterima lebih mudah dibaca dan diparsing disisi *client*.

Dalam pengaplikasiannya, REST lebih banyak digunakan untuk web Service yang berorientasi pada *resource*. Maksud orientasi pada *resource* adalah orientasi yang menyediakan *resource-resource* sebagai layanannya dan bukan kumpulan-kumpulan dari aktifitas yang mengolah *resource* itu. Beberapa contoh web *service* yang menggunakan REST adalah: Flickr API(Application ProgramInterface), YouTube API, Amazon API.

3.7. PHP

PHP adalah singkatan dari Bahasa Inggris yaitu Hypertext Prepocessor. Sedangkan pengertiannya adalah sebuah bahasa script atau pemograman web yang digunakan secara luas dan digunakan bersamaan dengan bahasa pemograman HTML (*Hyper Text Markup Language*) untuk membuat konten situs web yang dinamis. Untuk membangun website yang dinamis biasanya juga dibutuhkan juga MySQL dan atau phpmyadmin.

Pada awalnya PHP merupakan singkatan dari Personal Home Page yang pertama kali dibuat oleh Rasmus Rerdorf pada tahun 1995. Namun mulai versi 3.0 singkatan PHP berubah menjadi Hypertext Preprocessor yang diperkenalkan oleh perusahaan yang bernama Zend, yang membawa perubahan pada bahasa pemrograman php menjadi lebih baik, lebih bersih dan lebih cepat.

3.8. Postman

Postman adalah sebuah aplikasi *HTTP client* yang merupakan *plugin* dari *browser* Chrome. Fungsi Postman adalah untuk pengecekan *web service.* Postman dapat menampilkan hasil dari *HTTP request* yang kompleks sekalipun dengan cepat.

3.9. *Organization Chart*

Organization chart adalah diagram yang menunjukkan struktur organisasi dan hubungan dan jajaran relatif bagian dan posisi

/ pekerjaan. Istilah ini juga digunakan untuk diagram yang sama, misalnya yang menampilkan unsur-unsur yang berbeda dari bidang pengetahuan atau kelompok bahasa.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

Pada kerja praktik kali ini, didapatkan 2 tugas yang berbeda, yang pertama adalah analisa dan perancangan system *organization chart* gtHR, sedangkan yang kedua adalah analisa dan perancangan *web service* sistem simpleCRM. Pada bab ini akan dijelaskan megenai hasil analisis dan rekomendasi yang diberikan kepada perusahaan terkait dengan kedua proyek yang sudah dikerjakan.

4.1. *Analisis Sistem organization chart GtEnterprise Human Resources (gtHR)*

4.1.1. Deskripsi Umum Aplikasi

GtEnterprise Human Resources atau disingkat gtHR adalah *software* yang memberikan kemudahan dalam pengelolaan data karyawan, pencatatan proses mutasi, penilaian kerja, penyederhanaan pengelolaan gaji dan penghitungan PPh (Pajak Penghasilan) serta melakukan rekapitulasi kehadiran karyawan. gtEnterprise HR dibangun dengan tujuan agar pengelola perusahaan mengetahui kondisi karyawan dan pada akhirnya dapat mengambil keputusan yang tepat terkait dengan strategi SDM yang akan di terapkan pada perusahaan tersebut.

Software gtHR dirancang agar dapat mengelola *database* presensi karyawan, baik yang dilakukan dengan sistem *barcode*, *fingerprint* dan *smartcard*, dan kemudian mengintegrasikannya dengan sistem payroll karyawan. Kerumitan dalam penghitungan gaji dan mengetahui kapan karyawan habis kontrak, dapat diselesaikan dengan aplikasi ini.

4.1.2. Analisis Fungsionalitas Aplikasi

Secara umum spesifikasi kebutuhan pengguna dari aplikasi yang akan dibangun meliputi:

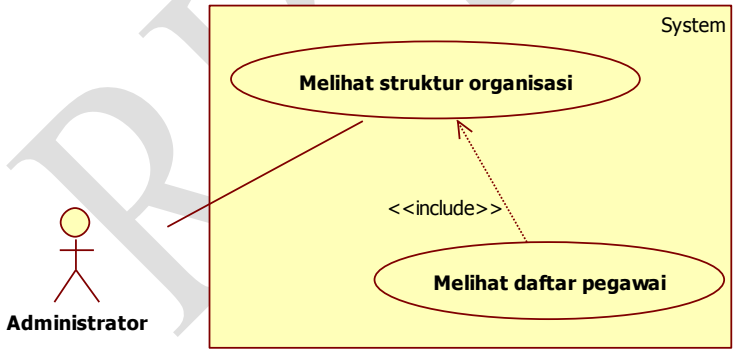
a) **Menampilkan Struktur organisasi dalam bentuk *organization chart***
Menampilkan setiap jabatan yang ada pada struktur organisasi sebagai alternatif dari tampilan berbentuk table.

b) Menampilkan daftar pegawai pada *organization chart*
Menampilkan daftar nama beserta dengan jabatan dari sebuah posisi yang dipilih.

4.2. Perancangan Sistem *organization chart* GtEnterprise Human Resources (gtHR)

4.2.1. Perancangan Fungsionalitas Aplikasi

Gambaran spesifikasi kebutuhan Sistem GtEnterprise *Human Resources* (gtHR) dapat digambarkan dengan diagram kasus penggunaan di bawah ini.



Gambar 4.1 Diagram Kasus *organization chart* GtEnterprise *Human Resources (gtHR)*

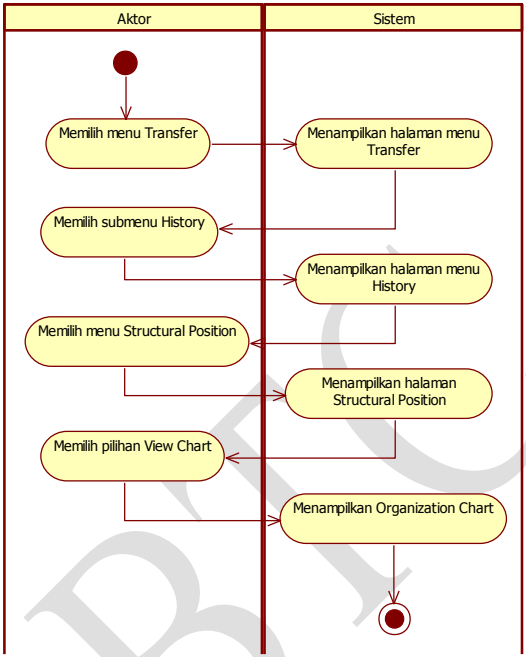
4.2.2. Cara Kerja Aplikasi

Fitur ini dikembangkan untuk menyediakan tampilan struktur organisasi berbentuk bagan.

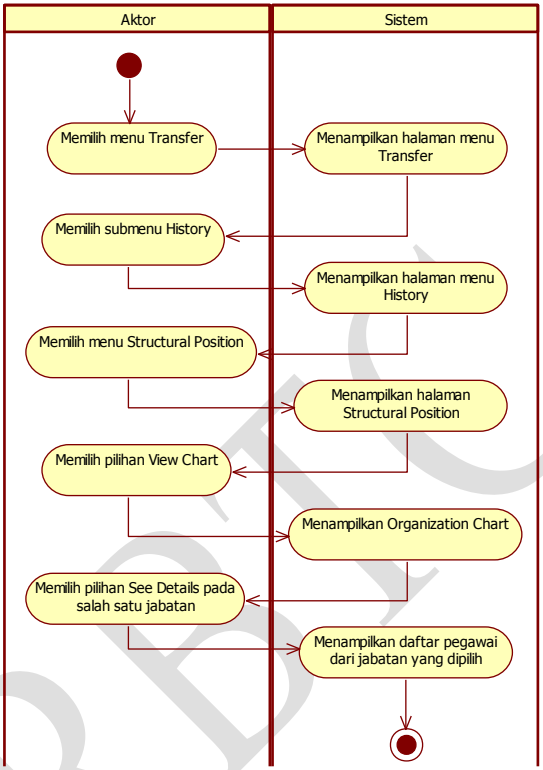
1. **Menampilkan Struktur organisasi dalam bentuk *organization chart***
   Administrator dapat melihat struktur organisasi dalam bentuk bagan.
   [Spesifikasi Pengembangan yang terkait: Subbab 4.1.2 poin a]

2. **Menampilkan daftar pegawai pada *organization chart***
   Administrator dapat melihat daftar pegawai pada semua *node* di bagan struktur organisasi.
   [Spesifikasi Pengembangan yang terkait: Subbab 4.1.2 poin b]

Untuk melengkapi penjelasan di atas, dokumen ini dilengkapi dengan diagram aktivitas untuk memperlihatkan alur dari setiap kasus penggunaan di atas.



Gambar 4.2 Diagram aktifitas Menampilkan Struktur organisasi dalam bentuk *organization chart*

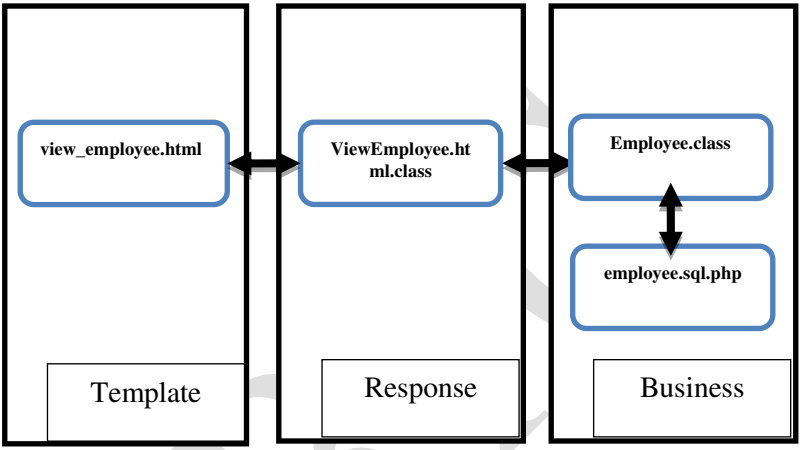


Gambar 4.3 Diagram aktifitas Menampilkan daftar pegawai pada *organization chart*

4.2.3. Arsitektur Perancangan Aplikasi

Secara keseluruhan, perangkat lunak yang dibangun dengan menggunakan GTFW akan memiliki arsitektur HMVC (*Hierarchy Model View Controller*) karena adanya pembagian dalam modul-modul. Dalam GTFW, '*Model'* disebut dengan '*Business'*, '*View'* disebut dengan '*Template'*, dan '*Controller'* disebut dengan '*Response'*.

Sistem ini menggunakan *model* view_empolee.html, response viewployee.htmll.class.php, dan business employee.class.php serta employee.sql.php. Gambaran arsitektur lengkapnya ditunjukkan oleh Gambar 4.4.



Gambar 4.4 *Sistem* organization chart *GtEnterprise Human Resources (gtHR)*

4.3. Analisis Sistem web service Simple Customer Relationship Management (SimpleCRM)

4.3.1. Deskripsi Umum Aplikasi

SimpleCRM atau simple adakah aplikasi yang memudahkan pengelolaan *customer*, mitra bisnis dan prospek perusahaan, memberikan pertimbangan kepada eksekutif perusahaan terkait proyek yang ada dalam perusahaan.

4.3.2. Analisis Kebutuhan Aplikasi

Secara umum spesifikasi kebutuhan pengguna dari aplikasi yang dibangun meliputi:

a. Menyediakan mekanisme *Login*
Menyediakan mekanisme untuk proses autentifikasi masuk kedalam sistem simpleCRM.

b. Melihat daftar pengguna
Menampilkan semua data pengguna yang terdaftar dalam database aplikasi.

c. Melihat data seorang pengguna
Menampilkan data lengkap dari pengguna yang dicari.

d. Mengubah data pengguna
Mengubah data pengguna yang sudah ada didalam database aplikasi.

e. Melihat daftar *client*
Menampilkan data *client* yang sudah tersedia dalam database aplikasi.

f. Menambah data *contact*
Menambahkan data *contact* untuk disimpan kedalam database aplikasi.

g. Menambah data *client*
Menambahkan data *client* untuk disimpan kedalam database aplilkasi.

h. Melihat daftar *Position*
Menampilkan daftar posisi dari pegawai.

i. Melihat daftar *sector*
Menampilkan daftar sektor atau divisi yang ditempati pegawai.

j. Melihat daftar *contact*
Menampilkan daftar data *contact* yang sudah tersimpan.

k. Melihat data *contact*
Menampilkan data sebuah kontak yang lengkap.

l. Melihat rangkuman data *lead*
Menampilkan data rangkuman dari proyek, semisal: total proyek yang aktif, total proposal yang aktif, dan total proyek yang di setujui.

m. Menambah data *activities*

Menambahkan data aktifitas terkait dengan proyek yang sedang dikerjakan..

n. Melihat data *lead*

Menampilkan data *lead* atau proyek yang sedang dikerjakan.

o. Melihat data *performance*

Menampilkan data statistik dari pegawai yang menjabat sebagai Account Executive dan per divisi.

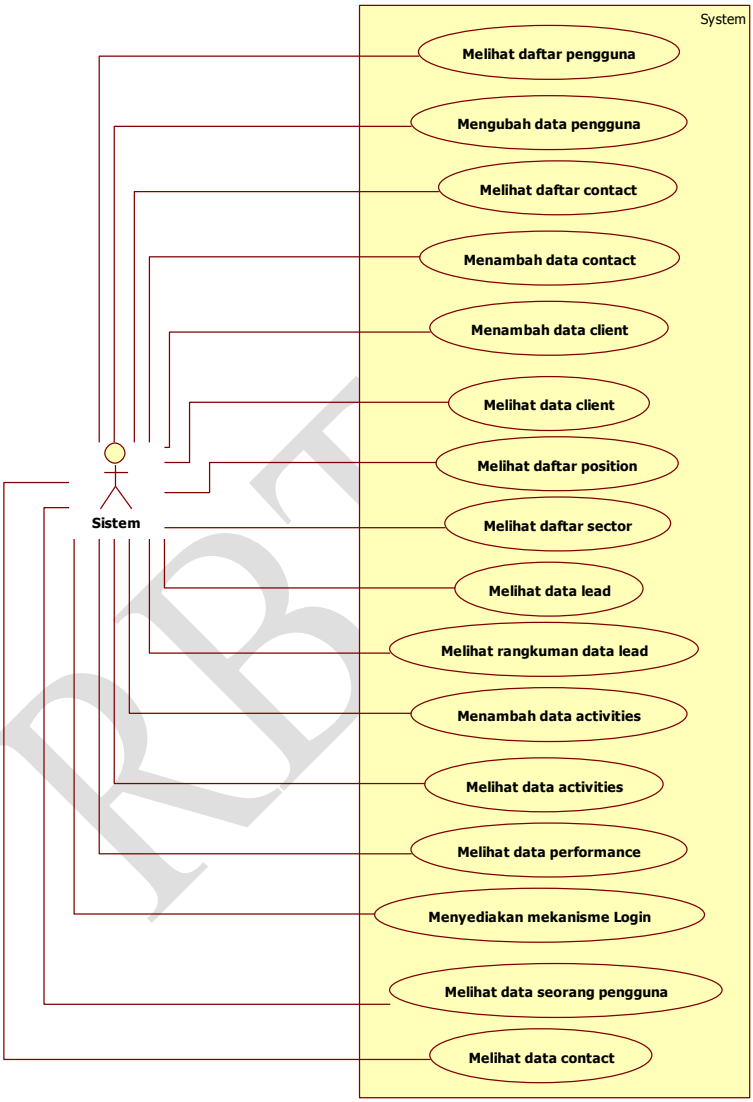
p. Melihat data tanggal aktifitas

Menampilkan data tanggal dari aktifitas dengan parameter *id user*.

4.4. Perancangan Sistem web service Simple Customer Relationship Management (SimpleCRM)

4.4.1. Perancangan Fungsionalitas Aplikasi

Gambaran spesifikasi kebutuhan dapat digambarkan dengan diagram kasus penggunaan di bawah ini.



Gambar 4.5 Digram kasus *Sistem web service Simple Customer Relationship Management (SimpleCRM)*

4.4.2. Cara Kerja Aplikasi

Sistem web *service* ini memfasilitasi pertukaran data dengan keluaran berupa *json.*

1. **Menyediakan mekanisme Login**
   Sistem melakukan proses autenfikasi dengan mencocokkan data *username* dan *password* dengan *record database*.
2. **Melihat daftar pengguna**
   Sistem melakukan *request* daftar pengguna ke database. Sistem akan menerima data daftar pengguna 10 *records* dalam satu kali *request*.
3. **Melihat data seorang pengguna**
   Sistem meminta data lengkap seorang pengguna dengan parameter berupa *id* pengguna.
4. **Mengubah data pengguna**
   Sistem melakukan *request* pengubahan data pengguna dengan parameter *id.* Apabila *id* yang diminta cocok dengan data di *database*, maka pengubahan data dapat dilakukan. Namun jika *id* tidak cocok, sistem akan mengirimkan pesan bahwa pengubahan data gagal dilakukan.
5. **Melihat daftar *client***
   Sistem melakukan *request* data daftar *client* ke database. Sistem akan menerima data daftar *client.*
6. **Menambah data *contact***
   Sistem melakukan *request* penambahan data *contact.* Apabila data yang ditambahkan sudah ada di *database*, maka penambahan data tidak dapat dilakukan dan sistem akan mengirimkan pesan gagal. Namun jika belum ada, sistem akan mengirimkan pesan bahwa penambahan data berhasil.
7. **Menambah data *client***
   Sistem melakukan *request* penambahan data *client.* Apabila data yang ditambahkan sudah ada di

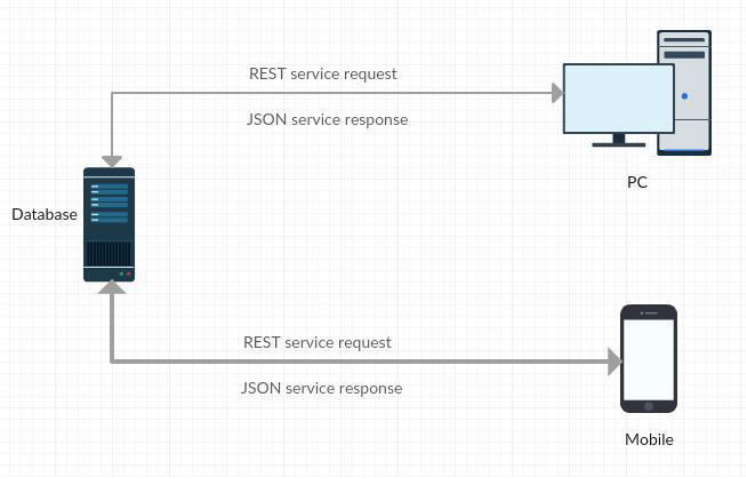
database, maka penambahan data tidak dapat dilakukan dan sistem akan mengirimkan pesan gagal. Namun jika belum ada, sistem akan mengirimkan pesan bahwa penambahan data berhasil.

8. **Melihat daftar *Position***
   Sistem melakukan *request* data *position* ke database. Sistem akan menerima data *position* 10 *records* dalam satu kali *request*.
9. **Melihat daftar *sector***
   Sistem melakukan *request* data *sector* ke *database*. Sistem akan menerima data *sector.*
10. **Melihat daftar *contact***
    Sistem melakukan *request* data daftar *contact* ke *database*. Sistem akan menerima data daftar pengguna 10 *records* dalam satu kali *request*.
11. **Melihat data *contact***
    Sistem melakukan *request* data lengkap sebuah *contact* ke *database.* Parameter yang dikirimkan adalah *id.*
12. **Melihat rangkuman data *lead***
    Sistem melakukan *request* data *lead* dengan parameter *id.* Apabila *id* yang diminta cocok dengan data di database, maka data akan ditampilkan. Namun jika *id* tidak cocok, sistem akan mengirimkan pesan bahwa pengambilan data gagal dilakukan.
13. **Menambah data *activities***
    Sistem melakukan *request* penambahan data *activities.* Apabila data yang ditambahkan sudah ada di *database*, maka penambahan data tidak dapat dilakukan dan sistem akan mengirimkan pesan gagal. Namun jika belum ada, sistem akan mengirimkan pesan bahwa penambahan data berhasil.

14. **Melihat data *lead***
    Sistem melakukan *request* data *lead* ke database. Sistem akan menerima data *lead.*
15. **Melihat data *performance***
    Sistem melakukan *request* data *performance* ke database. Sistem akan menerima data *performance.*
16. **Melihat data tanggal aktifitas**
    Sistem melakukan *request* data tanggal aktifitas dengan parameter berupa *id user*. Data yang akan diterima berupa *createdata*,*duedate*, dan *donedate*.

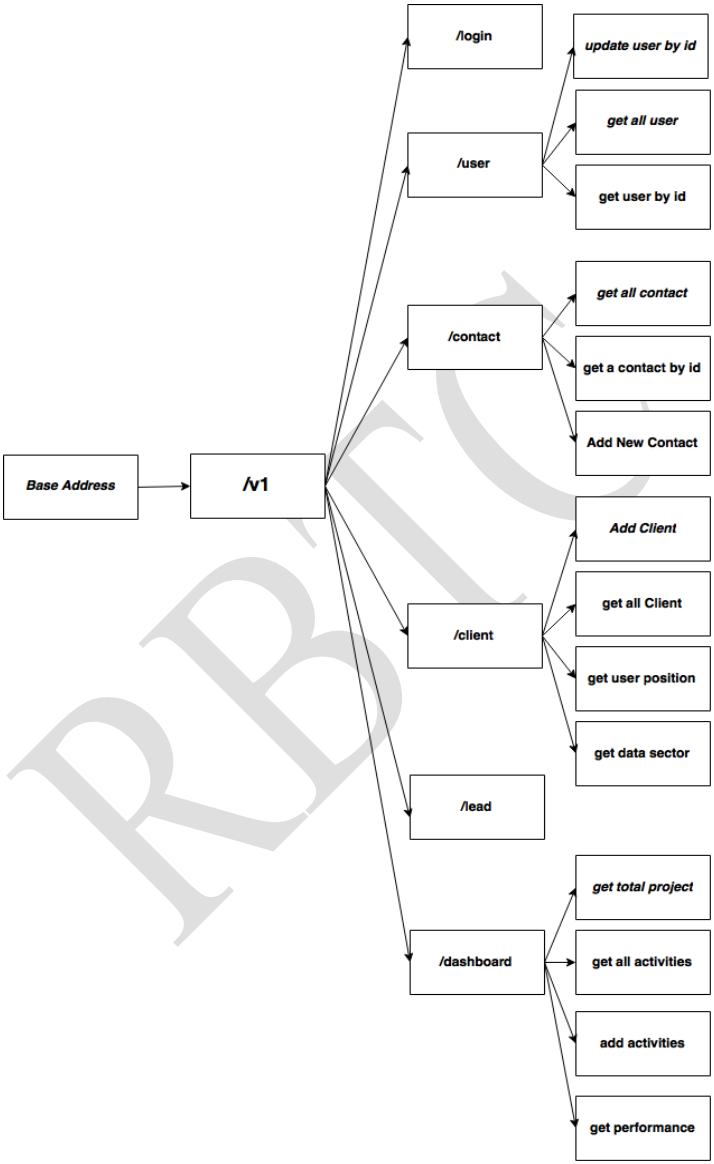
4.4.3. Arsitektur Perancangan Aplikasi

Gambaran umum arsitektur sistem *web service* simpleCRM dapat dilihat pada Gambar 4.6. Sistem meminta *request* kepada sistem *web service,* selanjutnya sistem *web service* mengambil atau menyimpan data ke *database*.



Gambar 4.6 Arsitektur *web service* simpleCRM

Untuk mengatur bagimana sistem *web service* diakses, maka dilakukan *routing*. *Routing* sistem *web service* ini ditunjukkan oleh Gambar 4.7 Routing sistem web service simpleCRM. *Routing* tersebut juga dapat membantu pengembang yang ingin menggunakan *web service* simpleCRM.



Gambar 4.7 Routing sistem web service simpleCRM

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Bahasa pemrograman yang digunakan adalah php.

5.1. Implementasi Sistem *organization chart GtEnterprise Human Resources (*gtHR)

Implementasi dari sistem ini dilakukan berdasarkan rekomendasi yang kami berikan sebelumnya pada tahap analisis. Sistem ini juga tidak memerlukan perancangan database, karena database yang sudah tersedia dapat digunakan tanpa perlu perbaikan. Selanjutnya kode implementasi akan dijelaskan persetiap rekomendasi yang telah diberikan.

5.1.1. Menambahkan fitur *organizational chart*

Fitur ini dikembangkan dengan bantuan JQuery. Data yang telah tersedia sebelumnya ditampilkan dalam bentuk tabel. Maka untuk implementasi fitur ini tinggal mengambil data yang tersedia kemudian mengolahnya membentuk *chart*.

a) Implementasi Lapisan Antarmuka

Lapisan ini berfungsi untuk membuat tampilan yang berdasarkan data yang berasal dari lapisan kontrol dan data. Sistem ini menggunakan view_employee.html untuk menampilkan data pegawai dan strukturnya dalam bentuk tabel dan *organization chart.* Kode lengkapnya dapat dilihat pada lampiran Kode sumber 6.

b) Implementasi Lapisan Kontrol

Pada lapisan kontrol ini digunakan untuk memanggil kelas *template* dan *business.* Terdapat tiga fungsi utama, yakni ProcessRequest(), TemplateModule(), dan ParseTemplate(). Fungsi ProcessRequest() digunakan sebagai jembatan dari *view* dan *controller* yang mengakses lapisan *business.* Fungsi TemplateModule()

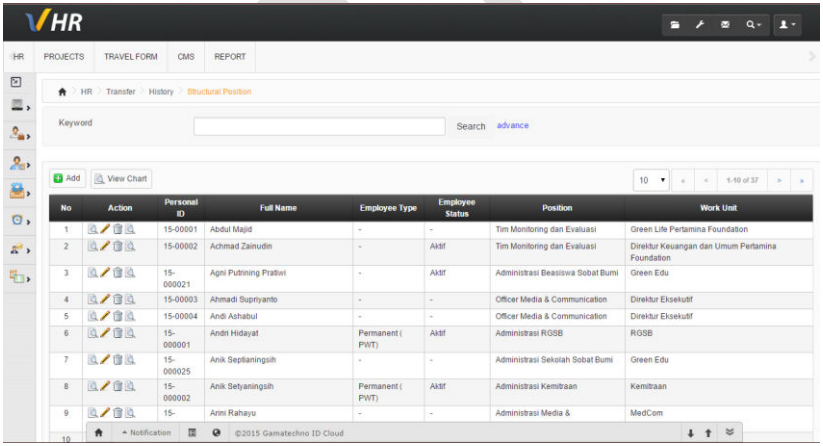
memanggil template yang digunakan, yakni view_employee.html. Sedangkan fungsi ParseTemplate() digunakan untuk mengirim data yang diolah ke lapisan antarmuka atau *template.* Kode lengkapnya ditunjukkan oleh lampiran Kode sumber 47.

c) Implementasi Lapisan Data

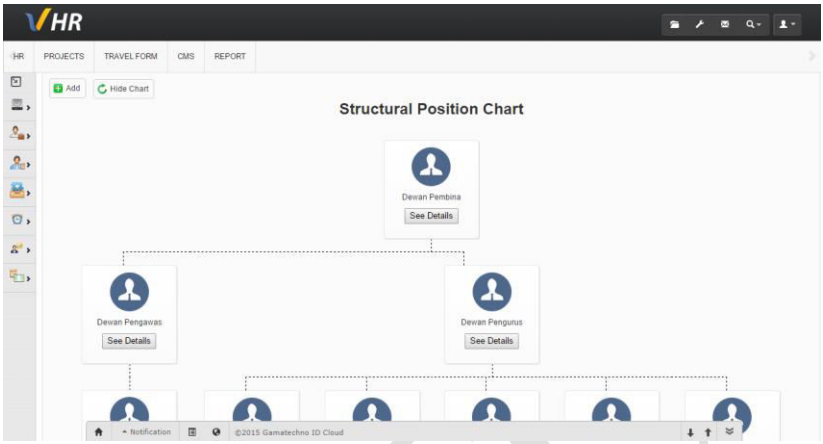
Data diambil oleh fungsi getStructurOrganization() yang berada pada model Employee.Class.php. Fungsi tersebut mengakses data dari *database* dengan cara memanggil implementasi *query* pada model Employee.sql.php. kode lengkapnya dapat dilihat pada lampiran Kode sumber 32 dan Kode sumber 33.

d) Implementasi Antarmuka Pengguna

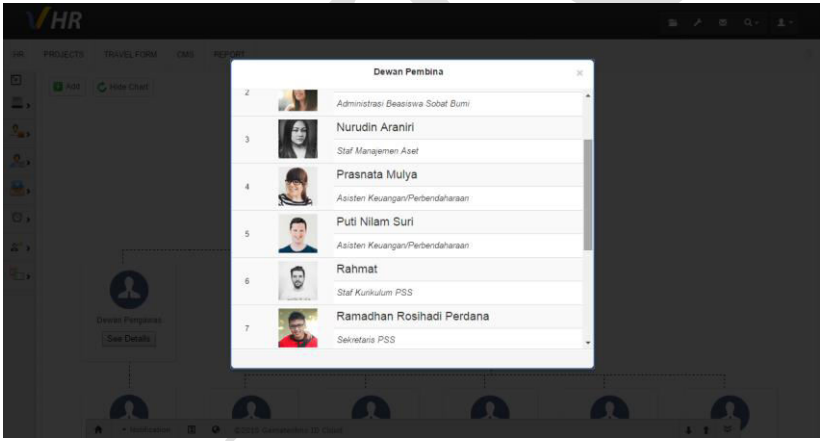
Tampilan dari implementasi *Sistem* organization chart GtEnterprise Human Resources (*gtHR)* ditunjukkan oleh Gambar 5.1, Gambar 5.2, dan Gambar 5.3.



Gambar 5.1 Tampilan halaman Structural Position pada gtHR



Gambar 5.2 Tampilan *Organizaton Chart*



Gambar 5.3 Tampilan daftar pegawai pada Tampilan *Organizaton Chart*

5.2. Implementasi Sistem Web Service Simple Customer Relationship Management (SimpleCRM)

Untuk implementasi pengembangan sistem *web service* ini, tidak perlu merancang skema basis data karena telah menggunakan skema basis data pada sistem yang lama. Antarmuka pengguna juga tidak perlukan karena web service ini hanya akan diakses oleh aplikasi, bukan oleh pengguna secara langsung. Selanjutnya, kode implementasi akan dijelaskan per setiap fungsional aplikasi.

5.2.1. Menyediakan mekanisme Login

Menggunakan metode *post* untuk bertukar data. Data yang dikirimkan adalah *username* dan *password*. Selanjutnya, data yang dirimkan sebelumnya akan dicocokkan dengan data yang ada di tabel *user*. Jika data pengguna yang dicari tersedia, maka login berhasil. Jika tidak, maka login dinyatakan gagal. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 29.

5.2.2. Melihat daftar pengguna

Menggunakan metode *get* untuk mengambil data. Data berasal dari tabel *user.* Memiliki fitur *pagination* yang berguna untuk membatasi pengiriman data, dalam hal ini dibatasi 10 *record* dalam satu *request.* Jika data tersedia, maka data pengguna akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk json. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 30.

5.2.3. Melihat data seorang pengguna

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan adalah *id.* Pertama, data *id* yang dirimkan sebelumnya akan di cocokkan dengan data yang ada di tabel *user.* Jika data tersedia,

maka data pengguna akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk json. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 31.

5.2.4. Mengubah data pengguna

Menggunakan metode *put* untuk bertukar data. Parameter yang digunakan adalah *id.* Pertama, data *id* yang dirimkan sebelumnya akan dicocokkan dengan data yang ada di tabel *user.* Jika data tersedia, maka data pengubahan data dapat dilakukan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 32.

5.2.5. Melihat daftar *contact*

Menggunakan metode *get* untuk bertukar data. Pertama, data akan ditambil dari tabel *client_cp* dan data asal tabel *client.* Kemudian data dicocokkan dan diambil data *client_name.* Data nomor *handphone* diambil dari tabel *client_cp_phone* dan data jabatan dari tabel *client_cp_position.* Terakhir, semua data yang dibutuhkan dimasukkan kedalam sebuah *array* dan dikirimkan dalam bensok *json.* Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 38.

5.2.6. Melihat data *contact*

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan adalah *id.* Pertama, data *id* yang dirimkan sebelumnya akan dicocokkan dengan data yang ada di tabel *client_cp.* Selanjutnya, data – data diambil dari tabel *client, client_cp_phone, client_cp_position.* Jika data *id* yang dikirimkan tersedia pada tebel *client_cp,* maka data kontak akan dikirimkan.

Jika tidak, maka pesan pemberitahuan bahwa kontak tidak tersedia akan dikimkan. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 39.

5.2.7. Menambahkan Data *Contact*

Menggunakan metode *post* untuk bertukar data. Pertama, data yang akan ditambahkan ditampung pada sebuah *array.* Lalu diperikasa apakah data tersebut sudah tersedia didalam *tabel client_cp.* Jika data belum tersedia, data dapat ditambahkan kedalam database. Jika sudah, maka sistem akan memberi pesan pemberitahuan bahwa data gagal ditambahkan kedalam *database.* Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 34.

5.2.8. Menambahkan Data *Client*

Menggunakan metode *post* untuk bertukar data. Pertama, data yang akan ditambahkan diperiksa kelengkapanya. Jika data tidak lengkap, maka operasi dianggap gagal. Jika data lengkap, maka selanjutnya data dicek apakah sudah tersedia ditabel *client.* Jika belum ada yang tersedia, maka data bisa ditambahkan jika data sudah tersedia maka, maka sistem akan memberi pesan pemberitahuan bahwa data gagal ditambahkan kedalam *database.* Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 35.

5.2.9. Melihat daftar *client*

Menggunakan metode *get* untuk mengambil data. Data berasal dari tabel *client.* Memiliki fitur *pagination* yang berguna untuk membatasi pengiriman data, dalam hal ini dibatasi 10 *record* dalam satu *request.* Jika data tersedia, maka data pengguna akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak

ditemukan. Balasan dari sistem dikirimkan dalam bentuk json. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 33.

5.2.10. Melihat daftar *Position*

Menggunakan metode *get* untuk mengambil data. Data berasal dari tabel c*lient_cp_position.* Memiliki fitur *pagination* yang berguna untuk membatasi pengiriman data, dalam hal ini dibatasi 10 *record* dalam satu *request.* Jika data tersedia, maka data pengguna akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 36.

5.2.11. Melihat daftar *sector*

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan adalah nama sektor. Data berasal dari tabel c*lient_sector.* Jika data tersedia, maka data sektor akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk json. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 37.

5.2.12. Melihat rangkuman data *lead*

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan adalah *lead*. Data berasal dari tabel *lead.* Jika data tersedia, maka data sektor akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 40.

5.2.13. Melihat data *lead*

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan adalah *id*. Pertama, data dari tabel *project* diambil. Data *Signed Project* diambil dari tabel *project,* data *Active Lead* dan data *Active Proposal* diambil dari tabel *lead.* Data yang diambil kemudian diolah dan disimpan dala sebuah *array.* Balasan dari sistem dikirimkan dalam bentuk json. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 42.

5.2.14. Menambah data activities

Menggunakan metode *post* untuk berkirim data. Parameter yang digunakan adalah data aktifitas. Pertama, masukkan dari pengguna akan ditampung didalam sebuah *array*. Kemudian data dimasukkan ke dalam *database.* Berhasil atau tidaknya data disimpan akan diberitahu ileh sistem. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 41.

5.2.15. Melihat data tanggal aktifitas

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan adalah *id.* Pertama, *id* yang diterima disimpan kedalam sebuah *array.* Kemudian data diambil dari tabel *activities.* Jika data tersedia, maka data akitifitas akan dikirimkan. Jika tidak tersedia, akan ada pemberitahuan bahwa pengguna tidak ditemukan. Balasan dari sistem dikirimkan dalam bentuk *json*. Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 44.

5.2.16. Melihat data performance

Menggunakan metode *get* untuk mengambil data. Parameter yang digunakan berupa data *performance.* Pertama, data dari tabel *user* diambil dengan kondisi *role_id* adalah "ae" dan del = 0. Untuk mendapat data Performa AM, data *project_value* dari tabel *project* dijumlahkan. Untuk mendapat data Performa Segmen, data dari tabel *project* dan *client_sector* di *left join*. Data kemudian dikemas dengan menggunakan *json.* Kode sumber lengkap dapat dilihat pada lampiran Kode sumber 43.

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

6.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan kerja praktik ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Processor : Intel® Xeon *Processor*
Memory : 16.00 GB
Jenis *Device* : Komputer
Sistem Operasi : Linux Ubuntu 12.04 LTS
Browser : Google Chrome 43

6.2. Skenario Pengujian

6.2.1. Skenario Pengujian Sistem *Organization GtEnterprise Human Resources* (gtHR)

Aplikasi gtHR diuji melalui aplikasi demo berbasis web milik Gamatechno yaitu demo.eoviz.com/hrm/hr dengan akun admin yang diberikan oleh pembimbing lapangan maupun dari *server* lokal. Pengujian dilakukan pada masing-masing sub modul yang diperbaiki maupun yang baru ditambahkan, yaitu *organization chart.*

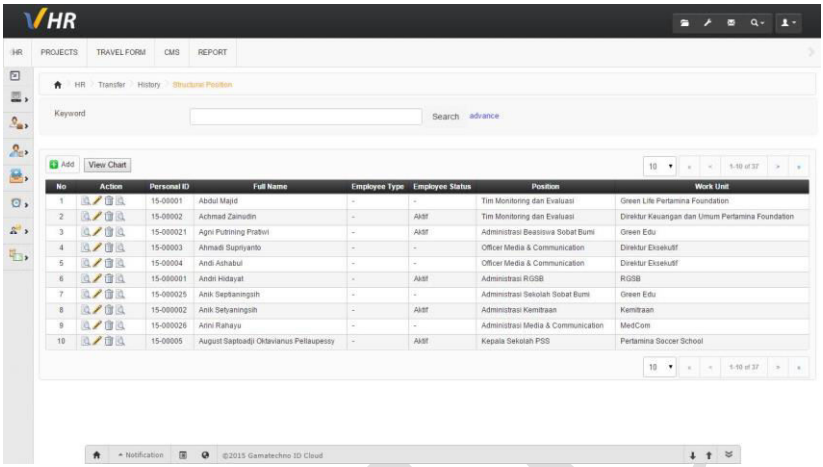
1. **Pengujian Menampilkan *Organizational Chart***
   Berikut adalah detil pengujian fungsionalitas menampilkan *organizational chart.*

Tabel 2 Pengujian fungsionalitas *organizational chart*

ID	KP-UJ.UC0001
Referensi Kasus Penggunaan	KP-UC0001
Nama	Pengujian fungsionalitas menampilkan *organization chart*

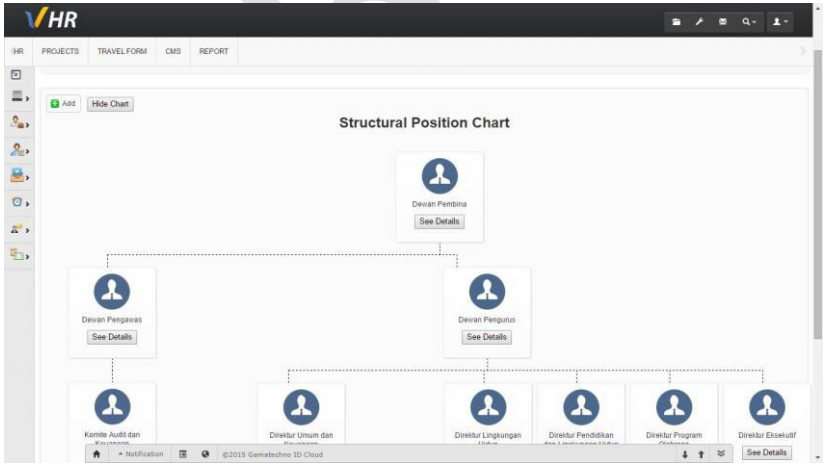
Tujuan Pengujian	Menguji fungsionalitas menampilkan *organization chart*
Skenario 1	Menampilkan *organization chart* pada halaman Structural Position
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel
Langkah Pengujian	1. Pengguna memilih menu Transfer → History → Structural Position 2. Pengguna memilih pilihan View Chart
Hasil Yang Diharapkan	Sistem menampilkan *organization chart*
Hasil Yang Didapat	Sistem menampilkan *organization chart*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan *organization chart*

Kondisi awal dari pengujian organizational chart ditampilkan pada Gambar 6.1.



Gambar 6.1 Kondisi awal pengujian fungsionalitas menampilkan *organizational chart*

Setelah memilih pilihan *View Chart*, maka akan ditampilkan *organizational chart* seperti yang ditunjukkan oleh Gambar 6.2.



Gambar 6.2 Kondisi akhir pengujian fungsionalitas menampilkan *organizational chart*

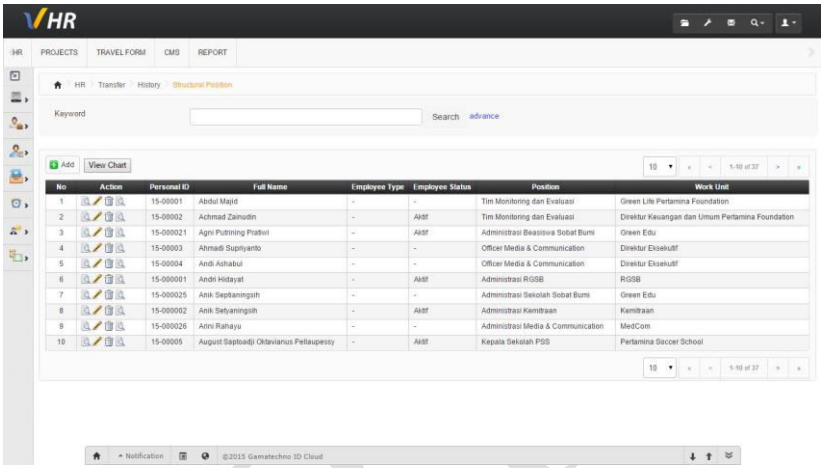
2. **Pengujian Menampilkan Daftar Pegawai pada *Organizational Chart***

   Berikut adalah detail pengujian fungsionalitas menampilkan daftar pegawai pada *organizational chart.*

Tabel 3 Pengujian fungsionalitas menampilkan daftar pegawai pada *organizational chart*

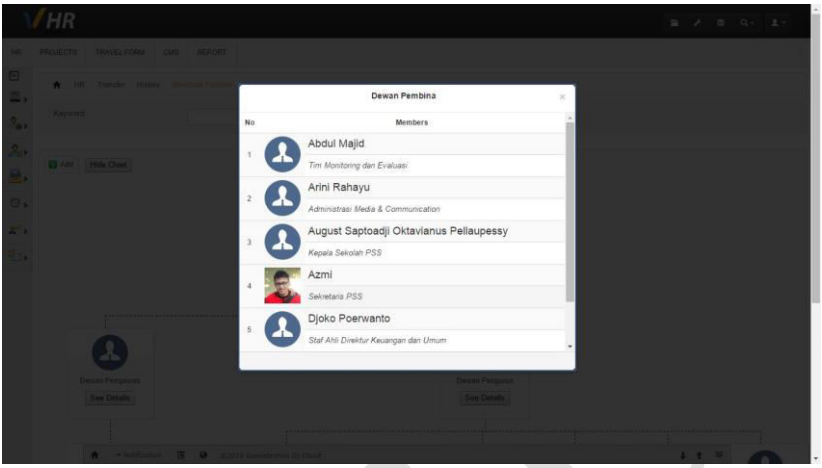
ID	KP-UJ.UC0002
Referensi Kasus Penggunaan	KP-UC0002
Nama	Pengujian fungsionalitas menampilkan daftar pegawai pada *organization chart*
Tujuan Pengujian	Menguji fungsionalitas menampilkan daftar pegawai pada *organization chart*
Skenario 1	Menampilkan daftar pegawai pada *organization chart* di halaman Structural Position
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel
Langkah Pengujian	1. Pengguna memilih menu *Transfer* → *History* → *Structural Position* 2. Pengguna memilih pilihan *View Chart* 3. Pengguna memilih pilihan *See Details* pada salah satu *node* jabatan yang dipilih
Hasil Yang Diharapkan	Sistem menampilkan daftar pegawai dari node jabatan yang dipilih
Hasil Yang Didapat	Sistem menampilkan daftar pegawai dari node jabatan yang dipilih
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan daftar pegawai dari node jabatan yang dipilih

Kondisi awal dari pengujian organizational chart ditampilkan pada Gambar 6.3.



Gambar 6.3 Kondisi awal pengujian fungsionalitas menampilkan daftar pegawai pada *organizational chart*

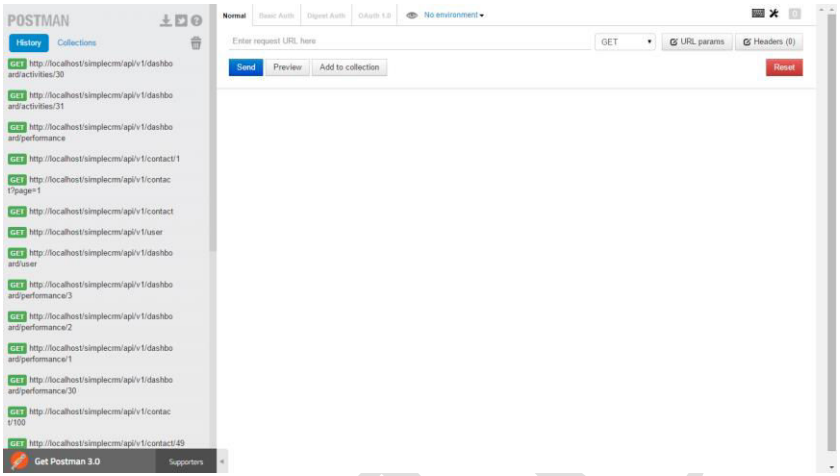
Setelah memilih pilihan *See Details*, maka akan ditampilkan daftar pegawai seperti yang ditunjukkan oleh Gambar 6.4.



Gambar 6.4 Kondisi akhir pengujian fungsionalitas menampilkan daftar pegawai pada *organizational chart*

6.2.2. Skenario Pengujian Sistem Simple Customer Relationship Management (SimpleCRM)

Pada pengujian ini, pada kami menggunakan ekstensi dari google chrome yakni Postman. Berikut adalah tampilan dari aplikasi postman ditunjukkan oleh Gambar 6.5.



Gambar 6.5 tampilan postman

1. **Pengujian API Menyediakan Mekanisme *login***
   Berikut adalah detail pengujian fungsionalitas API *Login.*

Tabel 4 Pengujian API Menyediakan Mekanisme *login*

ID	KP-UJ2.UC0101
Referensi Kasus Penggunaan	KP-UC0101
Nama	Pengujian API *Login*
Tujuan Pengujian	Memberikan keterangan status user setelah *login*
Skenario 1	Menampilkan data dan status user setelah *login*
Kondisi Awal	-

Data Uji	Data uji merupakan *record-record* pada tabel *user*
Parameter	username, password
End Point	POST /api/v1/login
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data dan status *user*
Hasil Yang Didapat	Sistem menampilkan data dan status *user*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data dan status *user*

Hasil dari pengujian 1 Menyediakan Mekanisme login ditampilkan pada Kode sumber 6


```
{
  "status": 200,
  "result": {
     "message": "Success",
     "id_user": "2",
     "fullname": "Aditya Nugraha",
     "role": "Direktur",
     "email": "direktur@gamatechno.com",
     "avatar": "1581414293879894884.jpg",
     "status": "logged in"
  }
```


Kode sumber 1 Pengujian API Menyediakan Mekanisme *login*

2. Pengujian API Melihat daftar pengguna

Berikut adalah detail pengujian fungsionalitas *API* Melihat daftar pengguna.

Tabel 5 Pengujian API melihat daftar pengguna

ID	KP-UJ2.UC0102
Referensi Kasus Penggunaan	KP-UC0102
Nama	Pengujian API *Get User Data*
Tujuan Pengujian	Mendapatkan data dari tabel user
Skenario 1	Menampilkan data user
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *user*
Parameter	*page*
End Point	GET /api/v1/user
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan Send
Hasil Yang Diharapkan	Sistem menampilkan data *user*
Hasil Yang Didapat	Sistem menampilkan data *user*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *user*

Hasil dari pengujian *API* Menyediakan Mekanisme login ditampilkan pada Kode sumber 3.

```
{
  "result": [
    {
      "id_user": "1",
      "username": "administrator"
    },
    {
      "id_user": "2",
      "username": "direktur"
    },
    {
      "id_user": "26",
      "username": "makmu"
    },
    {
      "id_user": "27",
      "username": "taufik"
    }
  ]
}
```

Kode sumber 2 Hasil Pengujian API Melihat daftar pengguna

3. Pengujian *API* Melihat data seorang pengguna

Berikut adalah detail pengujian fungsionalitas *API* Melihat data seorang pengguna.

Tabel 6 Pengujian Melihat data seorang pengguna

ID	KP-UJ2.UC0103
Referensi Kasus Penggunaan	KP-UC0103

Nama	Pengujian API Melihat data seorang pengguna
Tujuan Pengujian	Mendapatkan data user dengan *id* tertentu dari tabel user
Skenario 1	Menampilkan data user dengan *id* tertentu
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *user*
Parameter	*id*
End Point	GET api/v1/user/:id
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memasukkan parameter 3. Pengguna memilih pilihan Send
Hasil Yang Diharapkan	Sistem menampilkan data *user* dengan *id* yang diminta
Hasil Yang Didapat	Sistem menampilkan data *user* dengan *id* yang diminta
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *user* dengan *id* yang diminta

Hasil dari pengujian *API* Melihat data seorang pengguna ditampilkan pada Kode sumber 5.

```
{
   "result": {
      "fullname": "administrator",
      "phone": "123123",
      "email": "admin@gamatechno.com",
      "role_id": "adm",
      "last_login": "2015-04-28 09:22:24",
      "sector_id": ""
   },
   "status": 200
}
```


Kode sumber 3 Pengujian API Melihat data seorang pengguna

4. **Pengujian API Mengubah data pengguna**

Berikut adalah detail pengujian fungsionalitas *API* Mengubah data pengguna

Tabel 7 Pengujian API Mengubah data pengguna

ID	KP-UJ2.UC0104
Referensi Kasus Penggunaan	KP-UC0104
Nama	Pengujian API Mengubah data pengguna
Tujuan Pengujian	Mengubah data user dengan *id* tertentu dari tabel user
Skenario 1	Mengubah data user dengan *id* tertentu
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *user*

Parameter	*Fullname, email, phone*
End Point	PUT api/v1/user/:id
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memasukkan parameter 3. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan status dan pesan pengubahan
Hasil Yang Didapat	Sistem menampilkan status dan pesan pengubahan
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan status dan pesan pengubahan

Hasil dari pengujian *API* Mengubah data pengguna ditampilkan pada Kode sumber 7.

```
{
  "status": 200,
  "result": "Update success!"
}
```

Kode sumber 4 Pengujian *API* Mengubah data pengguna

5. Pengujian *API* Melihat daftar *client*

Berikut adalah detail pengujian fungsionalitas *API* Melihat daftar *client.*

Tabel 8 Pengujian *API* Melihat daftar *client*

ID	KP-UJ2.UC0105
Referensi Kasus Penggunaan	KP-UC0105
Nama	Pengujian API Melihat daftar *client*
Tujuan Pengujian	Menampilkan daftar *client*
Skenario 1	Menampilkan daftar *client*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client*
Parameter	*page*
End Point	GET api/v1/client
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan daftar *client*
Hasil Yang Didapat	Sistem menampilkan daftar *client*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan daftar *client*

Hasil dari pengujian API Melihat daftar *client* ditampilkan pada Kode sumber 9.

```
{
   "status": 200,
   "result": [
      {
         "id": "1",
         "client_name": "Pertamina Foundation ",
         "address": "Jl Sinabung 2  Simprug Jakarta Selatan"
      },
      {
         "id": "2",
         "client_name": "PT.Asia Afrika",
         "address": "Jakarta selatan"
      },
      {
         "id": "3",
         "client_name": "PT. GMUM",
         "address": "bulaksumur"
      },
      {
         "id": "4",
         "client_name": "Universitas Cendana",
         "address": "Manukwari Jayapura"
      }
   ],
   "page": "1"
}
```


Kode sumber 5 Pengujian API Melihat daftar *client*

6. Pengujian *API* Menambah data *contact*

Berikut adalah detail pengujian fungsionalitas *API* Menambah data *contact.*

Tabel 9 Pengujian *API* Menambah data *contact*

ID	KP-UJ2.UC0106
Referensi Kasus Penggunaan	KP-UC0106
Nama	Pengujian API Menambah data *contact*
Tujuan Pengujian	Menambahkan *contact*
Skenario 1	Menambahkan *contact*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client_cp*
Parameter	*last_name, last_name, email, phone_number[], position_id, company_id*
End Point	POST api/v1/contact
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memasukkan parameter 3. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan daftar status dan pesan penambahan
Hasil Yang Didapat	Sistem menampilkan daftar status dan pesan penambahan
Hasil Pengujian	Pengujian berhasil

Kondisi Akhir	Sistem menampilkan daftar status dan pesan penambahan

Hasil dari pengujian API Menambah data *contact* ditampilkan pada Kode sumber 6.

```
{
  "status": 200,
  "result": "Berhasil!"
}
```

Kode sumber 6 Pengujian *API* Menambah data *contact*

7. Pengujian *API* Menambah data *client*

Berikut adalah detail pengujian fungsionalitas *API* Menambah data *client.*

Tabel 10 Pengujian API Menambah data *contact*

ID	KP-UJ2.UC0107
Referensi Kasus Penggunaan	KP-UC0107
Nama	Pengujian API Menambah data *client*
Tujuan Pengujian	Menambahkan *client*
Skenario 1	Menambahkan *client*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client*

Parameter	*name, address, phone*
End Point	POST api/v1/client
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memasukkan parameter 3. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan daftar status dan pesan penambahan
Hasil Yang Didapat	Sistem menampilkan daftar status dan pesan penambahan
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan daftar status dan pesan penambahan

Hasil dari pengujian API Menambah data *client* ditampilkan pada Kode sumber 13.

```
{
  "status": 200,
  "result": "Berhasil!"
}
```

Kode sumber 7 Pengujian API Menambah data *contact*

8. Pengujian *API* Melihat daftar *Position*

Berikut adalah detail pengujian fungsionalitas *API* Melihat daftar *Position.*

Tabel 11 Pengujian *API* Melihat daftar *Position*

ID	KP-UJ2.UC0108
Referensi Kasus Penggunaan	KP-UC0108

Nama	Pengujian API Melihat daftar *Position*
Tujuan Pengujian	Menampilkan daftar *client position*
Skenario 1	Menampilkan daftar *client position*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client_cp_position*
Parameter	*page*
End Point	GET api/v1/client/position
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data *client position*
Hasil Yang Didapat	Sistem menampilkan data *client position*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *client position*

Hasil dari pengujian API Melihat daftar *Position* ditampilkan pada Kode sumber 15.


```
{
 "status": "200 OK",
 "result": [
  {
   "id_position": "1",
   "position": "President Director"
  },
  {
   "id_position": "2",
   "position": "Owner"
  },
  {
   "id_position": "3",
   "position": "Director - Operational"
  },
  {
   "id_position": "4",
   "position": "Director - Human Capital"
  },
  {
   "id_position": "5",
   "position": "Director - Finance"
  },
  {
   "id_position": "6",
   "position": "Manager - Operational"
  },
  {
   "id_position": "7",
   "position": "Manager - Human Capital"
```


Kode sumber 8 Pengujian *API* Melihat daftar *Position*

9. Pengujian *API Melihat daftar sector*

Berikut adalah detail pengujian fungsionalitas *API* Melihat daftar *sector.*

Tabel 12 Pengujian *API* Melihat daftar *sector*

ID	KP-UJ2.UC0109
Referensi Kasus Penggunaan	KP-UC0109
Nama	Pengujian API Melihat daftar *sector*
Tujuan Pengujian	Menampilkan daftar *client sector*
Skenario 1	Menampilkan daftar *client sector*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client_sector*
Parameter	-
End Point	GET api/v1/client/sector
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data *client sector*
Hasil Yang Didapat	Sistem menampilkan data *client sector*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *client sector*

Hasil dari pengujian API Melihat daftar *sector* ditampilkan pada Kode sumber 17.


```
{
 "status": "200 OK",
 "result": [
  {
   "id_sector": "1",
   "sector": "Academic"
  },
  {
   "id_sector": "2",
   "sector": "Goverment"
  },
  {
   "id_sector": "3",
   "sector": "Transportation"
  },
  {
   "id_sector": "4",
   "sector": "Lifestyle"
  },
  {
   "id_sector": "5",
   "sector": "asdf"
  },
```


Kode sumber 9 Pengujian *API Melihat daftar sector*

10. Pengujian *API* Melihat daftar *contact*

Berikut adalah detail pengujian fungsionalitas *API* Melihat daftar *contact.*

Tabel 13 Pengujian *API* Melihat daftar *contact*

ID	KP-UJ2.UC0110
Referensi Kasus	KP-UC0110

Penggunaan	
Nama	Pengujian API Melihat daftar *contact*
Tujuan Pengujian	Menampilkan daftar *contact*
Skenario 1	Menampilkan daftar *contact*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client, client_cp, client_cp_phone, client_cp_position*
Parameter	-
End Point	GET api/v1/contact
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data *contact*
Hasil Yang Didapat	Sistem menampilkan data *contact*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *contact*

Hasil dari pengujian *API* Melihat daftar *contact* ditampilkan pada Kode sumber 19.

```
{
   "status": 200,
   "data": [
      {
         "firstname": "Afrizal",
         "lastname": "Hernandar",
         "asal_pt": "PT. GMUM",
         "posisi": "Director - Operational",
         "email": "afrizal@gmum.com",
         "phone": [
            "11111111",
            "222222222"
         ]
      },
```

Kode sumber 10 Pengujian *API Melihat daftar contact*

11. Pengujian API Melihat data *contact*

Berikut adalah detail pengujian fungsionalitas API Melihat data *contact*

Tabel 14 Pengujian *API* Melihat data *contact*

ID	KP-UJ2.UC0111
Referensi Kasus Penggunaan	KP-UC0111
Nama	Pengujian API Melihat data *contact*
Tujuan Pengujian	Menampilkan data sebuah *contact* berdasarkan *id* yang dikirim.
Skenario 1	Menampilkan data *contact* dengan *id* tertentu

Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *client, client_cp, client_cp_phone, client_cp_position*
Parameter	-
End Point	GET api/v1/contact/:id
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data *contact* dengan id yang diminta
Hasil Yang Didapat	Sistem menampilkan data *contact* dengan id yang diminta
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *contact* dengan id yang diminta

Hasil dari pengujian *API* Melihat data *contact* ditampilkan pada Kode sumber 21.

```
{
  "result": {
     "firstname": "Endang",
     "lastname": "SSKH",
     "email": "endahsskh@gmail.com",
     "phone": "098765432",
     "jabatan": "Director - Finance",
     "asal": "Pertamina Foundation "
  },
  "status": 200
}
```

Kode sumber 11 Pengujian API Melihat data *contact*

12. Pengujian *API* Melihat rangkuman data *lead*

Berikut adalah detail pengujian fungsionalitas API Melihat rangkuman *data lead.*

Tabel 15 Pengujian *API* Melihat rangkuman data *lead*

ID	KP-UJ2.UC0112
Referensi Kasus Penggunaan	KP-UC0112
Nama	Pengujian API Melihat rangkuman data *lead*
Tujuan Pengujian	Menampilkan informasi *total signed project* , *total active lead* , dan *total active proposal*.
Skenario 1	Menampilkan data *dashboard total* dengan *id* tertentu.
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *user, project, client_sector,*
Parameter	*id*
End Point	GET dashboar/total/:id
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data-data *total signed project* , *total active lead* , dan *total active proposal* dengan *id* yang diminta
Hasil Yang Didapat	Sistem menampilkan data-data *total signed project* , *total active lead* , dan *total active proposal* dengan *id* yang diminta
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data-data *total signed project* , *total active lead* , dan *total active proposal* dengan *id*

	yang diminta

Hasil dari pengujian *API* Melihat rangkuman data *lead* ditampilkan pada Kode sumber 24 Pengujian *API* Melihat rangkuman data *lead*.


```
{
   "status": 200,
   "data": {
      "total_signed_project": {
         "total": 165000000,
         "from_project": 3
      },
      "total_active_lead": {
         "total": 500000000,
         "from_project": 1
      },
      "total_active_proposal": {
         "total": 0,
         "from_project": 0
      }
   }
}
```


Kode sumber 12 Pengujian *API* Melihat rangkuman data *lead*

13. Pengujian *API* Menambah *data activities*

Berikut adalah detail pengujian fungsionalitas API Menambah data *activities.*

Tabel 16 Pengujian *API* Menambah *data activities*

ID	KP-UJ2.UC0113
Referensi Kasus Penggunaan	KP-UC0113

Nama	Pengujian API Menambah data *activities*
Tujuan Pengujian	Menambahkan aktivitas user ke dalam *database*
Skenario 1	Menambahkan aktivitas user ke dalam *database*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *activities*
Parameter	*user_id, lead_id, due_date*
End Point	POST dashboard/activities
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan status dan pesan berhasil tidaknya penambahan data
Hasil Yang Didapat	Sistem menampilkan status dan pesan berhasil tidaknya penambahan data
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan status dan pesan berhasil tidaknya penambahan data

14. Pengujian *API* Melihat data *lead*

Berikut adalah detail pengujian fungsionalitas API Melihat data *lead*

Tabel 17 Pengujian *API* Melihat data *lead*

ID	KP-UJ2.UC0114
Referensi Kasus Penggunaan	KP-UC0114
Nama	Pengujian API Melihat data *lead*
Tujuan Pengujian	Menampilkan data *Lead* dari tabel *lead*
Skenario 1	Menampilkan data *Lead*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *lead*
Parameter	-
End Point	GET api/v1/lead
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data *Lead*
Hasil Yang Didapat	Sistem menampilkan data *Lead*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *Lead*

Hasil dari pengujian API Melihat data *lead* ditampilkan pada Kode sumber 25.

```
{
  "result": [
    {
      "id_lead": "1",
      "lead_title": "Aplikasi gtBeasiswa",
      "lead_description": "Aplikasi yang akan digunakan untuk mengelola data penerima beasiswa dan melakukan monitoring prestasi penerima beasiswa di Pertamina Foundation",
      "lead_currency": "0",
      "lead_value": "60000000",
      "client_id": "1",
      "cp_id": "1",
      "ae_id": "40",
      "il_id": "37",
      "date_created": "2015-03-20",
      "date_modified": "2015-03-20",
      "date_convert": "2015-04-28",
      "status": "4",
      "proposal_type": "0",
      "proposal_status": "1",
      "order": "3",
      "star": "0",
      "del": "0"
    }
  ],
  "status": 200
}
```


Kode sumber 13 Pengujian API Melihat data *lead*

15. Pengujian *API* Melihat data *performance*

Berikut adalah detail pengujian fungsionalitas API Melihat data *performance*

Tabel 18 Pengujian *API* Melihat data *performance*

ID	KP-UJ2.UC0114
Referensi Kasus Penggunaan	KP-UC0114
Nama	Pengujian API Melihat data *performance*
Tujuan Pengujian	Menampilkan data *Performance AM* dan *Performance segment*
Skenario 1	Menampilkan data *Performance*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel *user, project, client_sector*
Parameter	-
End Point	GET dashboard/performance
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data *Performance*
Hasil Yang Didapat	Sistem menampilkan data *Performance*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data *Performance*

Hasil dari pengujian API Melihat data *performance* ditampilkan pada Kode sumber 6.


```
{
  "data": {
    "performance_am": [
      {
        "name": "Agita",
        "avatar": "0412514294075413914.jpg",
        "total": "0"
      },
      {
        "name": "Dena Damayanti",
        "avatar": "4172214294075900039.jpg",
        "total": "0"
      },
      {
        "name": "Ririn",
        "avatar": "1476514294076163787.jpg",
        "total": "0"
      },
      {
        "name": "Anggun",
        "avatar": "4362914294076606804.jpg",
        "total": "0"
      },
      {
        "name": "Indah Putri",
        "avatar": "7327314294074670665.jpg",
        "total": "75000000"
      },
      {
        "name": "Sandy Wijaya",
```

```
            "avatar": "9458114294076855728.jpg",
            "total": "60000000"
        }
    ],
    "performance_segment": [
        {
            "segment": "Academic",
            "total": "0"
        },
        {
            "segment": "Goverment",
            "total": "0"
        },
        {
            "segment": "Transportation",
            "total": "0"
        },
        {
            "segment": "Lifestyle",
            "total": "165000000"
        }
    ]
},
"status": 200
```


Kode sumber 14 Pengujian *API* Melihat data *performance*

16. Pengujian *API* Melihat data tanggal aktifitas

Berikut adalah detail pengujian fungsionalitas API Melihat data tanggal aktifitas

Tabel 19 Pengujian *API* Melihat data tanggal aktifitas

ID	KP-UJ2.UC0114
Referensi Kasus Penggunaan	KP-UC0114
Nama	Pengujian API Melihat data tanggal aktifitas
Tujuan Pengujian	Menampilkan *create date, due date, dan done date dari tabel activities*
Skenario 1	Menampilkan data *Activities*
Kondisi Awal	-
Data Uji	Data uji merupakan *record-record* pada tabel
Parameter	*userid*
End Point	GET dashboard/activities
Langkah Pengujian	1. Pengguna memasukkan *end-point* ke URL *bar* Postman 2. Pengguna memasukkan parameter 3. Pengguna memilih pilihan *Send*
Hasil Yang Diharapkan	Sistem menampilkan data-data *Activities*
Hasil Yang Didapat	Sistem menampilkan data-data *Activities*
Hasil Pengujian	Pengujian berhasil
Kondisi Akhir	Sistem menampilkan data-data *Activities*

Hasil dari pengujian *API* Melihat data tanggal aktifitas ditampilkan pada Kode sumber 28.

```
{
   "data": {
      "createdata": "2015-03-30",
      "duedate": "2015-03-30 21:09:00",
      "donedate": "2015-03-30 21:13:00"
   },
   "status": 200
}
```


Kode sumber 15 Pengujian API Melihat data tanggal aktifitas

6.3. Evaluasi Pengujian

Hasil evaluasi dari uji coba sistem *organization chart* fungsinya telah berjalan dengan baik. Sistem sudah dapat menampilkan *organization chart* dan detail pegawai dengan baik. Sedangkan sistem *web service* simpleCRM sudah mampu memenuhi semua kebutuhan fungsionalnya dengan baik. *Web service* telah dapat digunakan untuk bertukar data.

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

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

7.1. Kesimpulan

Dengan hasil pengujian dan evaluasi, maka dapat diambil kesimpulan sebagai berikut:

1. *Administrator* dapat melihat struktur organisasi perusahaan dalam bentuk bagan yang dihasilkan oleh *jquery organizational chart* dengan cara mengakses menu *Structural Position*,
2. Pada menu *Structural Position*, daftar pegawai masing-masing jabatan dapat dilihat dengan memilih pilihan *See Details* pada masing-masing *node* jabatan yang ditampilkan oleh *organizational chart.*
3. Akses *database* dapat diperoleh dengan menggunakan fasilitas yang disediakan *web service*. *Administrator* bisa mendapatkan data dengan mengakses *endpoint* baik melalui aplikasi *desktop* maupun *mobile.* Hal ini dikarenakan keluaran *web service* berupa file *json* yang bisa mendukung berbagai *platform* terutama *desktop* dan *mobile*.

7.2. Saran

Pada kerja praktik ini, ada beberapa saran dari penulis agar pelaksanaan kerja praktik di Gamatechno selanjutnya bisa lebih baik. Berikut beberapa saran dari penulis:

1. Pada masa awal kerja praktik sebaiknya diadakan pelatihan singkat GTFW dari pihak Gamtechno. Tujuannya agar peserta kerja praktik bisa lebih cepat memahami penggunaan GTFW sehingga menghemat waktu pengerjaan tugas kerja praktik.
2. Tugas-tugas yang diberikan sebaiknya sesuai dengan kompetensi yang diminta oleh peserta kerja praktik.
3. *Monitoring* peserta kerja praktik sebaiknya lebih diperketat agar pelaksanaan kerja praktik bisa lebih terkontrol mengingat tempat kerja peserta kerja praktik terpisah dengan karyawan.

Demikian saran dari penulis. Semoga bisa menjadi evaluasi agar lebih baik ke depannya.

DAFTAR PUSTAKA

PT Gamatechno Indonesia. 2015. gtHR, http://gamatechno.com/products-services-detail/43-corporate-suite/254-gthr-2, (diakses 13 Juni 2015).

PT Gamatechno Indonesia. 2015. CRM, http://eoviz.com/software/softwareCRM, (diakses 13 Juni 2015).

CULESSHABRUR. 2015. REST API, http://s4nbao.blogspot.com/2013/01/rest-api.html, (diakses 20 Juni 2015).

PT Gamatechno Indonesia. 2015. GTFW, http://gtfw.gamatechno.com/, (diakses 15 Juni 2015).

Josh Lockhart. 2015. Slim, http://www.slimframework.com/, (diakses 15 Juni 2015).

PT Gamatechno Indonesia. 2015. Live up your consumer lifestyle, http://gamatechno.com/products-services/lifestyle-solution, (diakses 16 Juni 2015).

Faried Blogger. 2012. Pengertian Breadcrumb dan Manfaat Breadcrumb Untuk SEO, http://tempat-inspirasiku.blogspot.com/2012/03/pengertian-breadcrumb-dan-manfaat.html, (diakses 15 Juni 2015).

Faizal Indra K.. 2015. Melewati Batas Maksimal Upload di PHP, http://www.topunik.com/artikel/teknologi/melewati-batas-maksimal-upload-di-php/ blue, (diakses 18 Juni 2015).

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Lampiran Kode Sumber

```
$app->post("/login", function () use ($app, $db) {
        $input = $app->request()->post();
        $data = array("username" =>
$input['username'], "password" =>
md5($input['password']));
        $user = $db->user()->where($data);
        if ($data = $user->fetch()) {
            $users['status'] = 200;
            $users['result'] = array(
                "message" => "Success",
                "id_user" => $data["id_user"],
                "fullname" => $data["fullname"],
                "role" => getRole($data["role_id"]),
                "email" => $data["email"],
                "avatar" => $data['avatar'],
                "status" => "logged in");
            echo json_encode($users);
        }
        else {
            $users['status'] = 400;
            $users['result'] = array(
                    "error"=>"login failed",
                    "message" => "Wrong
password/username"
                );
            echo json_encode($users);
        }
    });


$app->post("/login", function () use ($app, $db) {
        $input = $app->request()->post();
        $data = array("username" =>
$input['username'], "password" =>
md5($input['password']));
        $user = $db->user()->where($data);
```

```
        if ($data = $user->fetch()) {
            $users['status'] = 200;
            $users['result'] = array(
                "message" => "Success",
                "id_user" => $data["id_user"],
                "fullname" => $data["fullname"],
                "role" => getRole($data["role_id"]),
                "email" => $data["email"],
                "avatar" => $data['avatar'],
                "status" => "logged in");
            echo json_encode($users);
        }
        else {
            $users['status'] = 400;
            $users['result'] = array(
                    "error"=>"login failed",
                    "message" => "Wrong
password/username"
                );
            echo json_encode($users);
        }
    });
```

Kode sumber 16 Menyediakan mekanisme Login

```
$app->get("/", function () use ($app, $db) {
$limit = $app->request()->get('page');
if(!isset($limit)) $limit = 1;
$offset = ($limit * 10) - 10;
$user = $db->user()->limit(10, $offset);
foreach ($user as $row) {
$users['result'][] = array("id_user" =>
$row['id_user'], "username" =>
$row['username']);
}
if (!$users) {
$users['status'] = 400;
$users['message'] = 'No data exists';
echo json_encode($users);
}
else {
$users['status'] = 200;
$users['page'] = $limit;
echo json_encode($users);
}
});

$app->get("/", function () use ($app, $db) {
$limit = $app->request()->get('page');
if(!isset($limit)) $limit = 1;
$offset = ($limit * 10) - 10;
$user = $db->user()->limit(10, $offset);
foreach ($user as $row) {
$users['result'][] = array("id_user" =>
$row['id_user'], "username" =>
$row['username']);
}
if (!$users) {
$users['status'] = 400;
$users['message'] = 'No data exists';
echo json_encode($users);
}
else {
```

```
$users['status'] = 200;
$users['page'] = $limit;
echo json_encode($users);
}
});
```

Kode sumber 17 Melihat daftar pengguna

```
$app->get("/:id", function ($id) use ($app,
$db) {
$data = array('id_user' => $id);
$user = $db->user()->where($data);
$count=0;
foreach ($user as $row) {
$users['result'] = array("fullname" =>
$row["fullname"], "phone" => $row["phone"],
"email" => $row["email"], "role_id" =>
$row["role_id"], "last_login" =>
$row["last_login"], "sector_id" =>
$row["sector_id"]);
$count++;
}
if ($count==0) {
$users['status'] = 400;
$users['result'] =  "Users does not exist";
echo json_encode($users);
}
else {
$users['status'] = 200;
echo json_encode($users);
}
});

if ($count==0) {
$users['status'] = 400;
$users['result'] =  "Users does not exist";
echo json_encode($users);
}
else {
$users['status'] = 200;
echo json_encode($users);
}
});
```

Kode sumber 18 Melihat data seorang pengguna

```
$app->put("/:id", function ($id)
use($app,$db){
            $data = array('id_user' => $id);
            $user = $db->user()-
>where("id_user", $id);
            if ($user->fetch()) {
                $input = $app->request()-
>put();
                $result = $user-
>update($input);
                if($result){
                    $users['status'] = 200;
                    $users['result'] = 'Update
success!';
                    echo json_encode($users);
                }else{
                    $users['status'] = 400;
                    $users['result'] = 'Update
failed!';
                    echo json_encode($users);
                }

            }
            else{
                //echo $user;
                $users['status'] = 400;
                $users['result'] = 'Update
failed!';
                echo json_encode($users);
            }
        });
```

Kode sumber 19 Mengubah data pengguna

```
$app->get("/", function () use ($app, $db) {
$limit = $app->request()->get('page');
if(!isset($limit)) $limit = 1;
$offset = ($limit * 10) - 10;
$client = $db->client()->limit(10, $offset);
if ($db->client()) $clients['status'] = 200;
$count = 0;

foreach ($client as $row) {
$clients['result'][] = array(
"id" => $row['id_client'],
"client_name" => $row['client_name'],
"address" => $row['client_address']
);
$count++;
}
if ($count == 0) {
$clients['message'] = 'No data exists';
echo json_encode($clients);
}
else {
$clients['page'] = $limit;
echo json_encode($clients);
}
});
```

Kode sumber 20 Melihat daftar client

```
$app->post("/", function () use ($app, $db) {
$input = $app->request()->post();
$contact = $db->client_cp();
$data = array(
"cp_firstname" => $input['first_name'],
"cp_lastname" => $input['last_name'],
"cp_email" => $input['email'],
"position_id" => $input['position_id'],
"client_id" => $input['company_id']
);

//Cek Apakah ada data yang double
$cek = $db->client_cp()->where($data);
$count=0;
foreach ($cek as $row) {
$count++;
};
if($count==0){
$phones = $app->request()->post("phone_number");
$count=0;
foreach ($phones as $newphone) {
$number = array("phone_number" => $newphone);
$cek = $db->client_cp_phone()->where($number);
foreach ($cek as $row) {
$count++;
};
}
if($count==0){
$result = $contact->insert($data);
$cp = $db->client_cp()->where($data);
$count=0;
foreach ($cp as $row) {
$cp_id = $row["id_cp"];
$count++;
};
//Add Multiple Phone Number
$phone = $db->client_cp_phone();
```

```
$phones = $app->request()->post("phone_number");
foreach($phones as $newphone){
$data2 = array(
"phone_number" => $newphone,
"phone_cp_id" => $cp_id
);
$result2 = $phone->insert($data2);
}

if($result2 && $result){
$message['status'] = 200;
$message['result'] = 'Berhasil!';
}
else{
$message['status'] = 400;
$message['result'] = 'Gagal!';
}
}
else{
$message['status'] = 400;
$message['result'] = 'Gagal!';
}
}
else{
$message['status'] = 400;
$message['result'] = 'Gagal!';
}
echo json_encode($message);
});
```

Kode sumber 21 Menambah data contact

```
$app->post("/", function () use ($app, $db) {
$input = $app->request()->post();
if($input['name']!=NULL &&
$input['address']!=NULL &&
$input['phone']!=NULL){
$client = $db->client();
$data = array(
"client_name" => $input['name'],
"client_address" => $input['address'],
"client_phone" => $input['phone']
);
//Cek Apakah ada data yang double
$cek = $db->client()->where($data);
$count=0;
foreach ($cek as $row) {
$count++;
};
if($count==0){
$result = $client->insert($data);
if($result){
$message['status'] = 200;
$message['result'] = 'Berhasil!';
}
else{
$message['status'] = 400;
$message['result'] = 'Gagal!';
}
}
else{
$message['status'] = 400;
$message['result'] = 'Gagal!';
}
}
else{
$message['status'] = 400;
$message['result'] = 'Gagal!';
}
echo json_encode($message);
});
```

Kode sumber 22 Menambah data client

```
$app->get("/position", function () use ($app,
$db) {
$limit = $app->request()->get('page');
if(!isset($limit)) $limit = 1;
$offset = ($limit * 10) - 10;
$client = $db->client_cp_position()->limit(10,
$offset);
foreach ($client as $row) {
$clients['result'][] = array(
"id_position" => $row["id_position"],
"position" => $row["position"]);
}
if (!$clients) {
$clients['status'] = 400;
$clients['message'] = 'Clients does not exists';
echo json_encode($clients);
}
else {
$clients['status'] = 200;
$clients['page'] = $limit;
echo json_encode($clients);
}
});
```

Kode sumber 23 Melihat daftar Position

```
$app->get("/sector", function () use ($app, $db)
{
            $client = $db->client_sector();
            $count = 0;

            foreach ($client as $row) {
                $clients['result'][] = array(
                    "id_sector" =>
$row["id_sector"],
                    "sector" => $row["sector"]);
                $count++;
            }

            if ($count == 0) {
                $clients['status'] = 400;
                $clients['message'] = 'Clients
does not exists';
                echo json_encode($clients);
            }
            else {
                $clients['status'] = 200;
                echo json_encode($clients);
            }
        });
```

Kode sumber 24 Melihat daftar *sector*

```
$app->get("/", function () use ($app, $db) {
$limit = $app->request()->get('page');
if(!isset($limit)) $limit = 1;
$offset = ($limit * 10) - 10;
$nama = $db->client_cp()->order("cp_firstname");
$message['status'] = 200;
$count=0;
foreach ($nama as $row) {
$asal = $db->client();
foreach ($asal as $row2) {
if($row2['id_client'] == $row['client_id']){
$company = $row2['client_name'];
}
};
$phones = $db->client_cp_phone();
$count3 = 0;
foreach ($phones as $row3) {
if($row3['phone_cp_id'] == $row['id_cp']){
$phone[$count3] = $row3['phone_number'];
$count3++;
}
};
$posisi = $db->client_cp_position();
foreach ($posisi as $row4) {
if($row4['id_position'] == $row['position_id']){
$position = $row4['position'];
}
};
$message['data'][] = array (
"id" => $row['id_cp'],
"firstname" => $row['cp_firstname'],
"lastname" => $row['cp_lastname'],
"asal_pt" => $company,
"posisi" => $position,
"email" => $row['cp_email'],
"phone" =>$phone);
$count++;
```

Kode sumber 25 Melihat daftar contact

```
$app->get("/:id", function ($id) use ($app, $db)
{$data = array('id_cp' => $id);
$contact = $db->client_cp()->where($data)-
>fetch();
$phone_number = $db->client_cp_phone()
->select("phone_number")
->where("phone_cp_id = ?", $id)
->fetch();
$jabatan = $db->client_cp_position()
->select("position")
->where("id_position = ?",
$contact["position_id"] )
->fetch();
$asal    =    $db->client()
->select("client_name")
->where("id_client = ?", $contact["client_id"])
->fetch();

if($contact != null){
$contacts['status'] = 200;
$app->response()->header("Content-Type",
"application/json");
$contacts['data'] = array(
"firstname" => $contact["cp_firstname"],
"lastname"  => $contact["cp_lastname"],
"email"     => $contact["cp_email"],
"phone"     => $phone number["phone number"],
"jabatan"   => $jabatan["position"],
"asal"      => $asal["client_name"]
);
echo json_encode($contacts);
}
else{
$app->response()->header("Content-Type",
"application/json");
$contacts['status'] = 400;
$contacts['result'] = "Users does not exist";
echo json encode($contacts);
```

Kode sumber 26 Melihat data contact

```
$app->get("/total/:id", function ($id) use ($app,
$db) {
if ($db->project()) $message['status'] = 200;
else $message['status'] = 400;
//Get Total Signed Project
$data = array(
"del" => 0
);
$project = $db->project()->where($data);
$signed_project_value = 0;
$signed_project_total = 0;
foreach ($project as $row) {
$signed_project_value = $signed_project_value +
$row['project_value'];
$signed_project_total++;
};
//Get Total Active Lead
$data2 = array(
"status" => 0,
"del" => 0
);
$lead2 = $db->lead()->where($data2);
$active_lead_value = 0;
$active_lead_total = 0;
foreach ($lead2 as $row2) {
$active_lead_value = $active_lead_value +
$row2['lead_value'];
$active_lead_total++;
};
//Get Total Active Proposal
$data3 = array(
"proposal_status" => 0,
"status" => 2,
"del" => 0
);
$lead = $db->lead()->where($data3);
$active_proposal_value = 0;
$active_proposal_total = 0;
foreach ($lead as $row) {
```

```
$active_proposal_value = $active_proposal_value +
$row['lead_value'];
$active_proposal_total++;
};

//Put Result into array of array
$total_signed_project = array(
"total" => $signed_project_value,
"from_project" => $signed_project_total
);
$total_active_lead = array(
"total" => $active_lead_value,
"from_project" => $active_lead_total
);
$total_active_proposal = array(
"total" => $active_proposal_value,
"from_project" => $active_proposal_total
);
$message['data'] = array(
"total_signed_project" => $total_signed_project,
"total_active_lead" => $total_active_lead,
"total_active_proposal" => $total_active_proposal
);
echo json_encode($message);
});
```

Kode sumber 27 Melihat rangkuman data lead

```
$app->post("/activities", function () use($app,
$db){
            $input = $app->request->post();
            //if(!isset($input['due_date']))
$input['due_date'] = date('Y-m-d h:i:s');
            $data = array(
                'user_id' => $input['user_id'],
                'lead_id' => $input['lead_id'],
                'due_date' => $input['due_date']
                );
            $result = $db->activities-
>insert($data);
            if($result)
            {
                $message['status'] = 200;
                $message['result'] = 'Insert
success';
            }
            else {
                $message['status'] = 400;
                $message['result'] = 'Insert
failed';
            }
            echo json_encode($message);
        });
```

Kode sumber 28 Menambah data activities

```
$app->get("/lead", function () use ($app, $db)  {
$lead = $db->lead();
$app->response()->header("Content-Type",
"application/json");
foreach ($lead as $row) {
$leads['result'][] = array("id_lead"
=>$row["id_lead"],
"lead_title" => $row["lead_title"],
"lead_description" => $row["lead_description"],
"lead_currency" => $row["lead_currency"],
"lead_value" => $row["lead_value"],
"client_id" => $row["client_id"],
"cp_id" => $row["cp_id"],
"ae_id" => $row["ae_id"],
"il_id" => $row["il_id"],
"date_created" => $row["date_created"],
"date_modified" => $row["date_modified"],
"date_convert" => $row["date_convert"],
"status" => $row["status"],
"proposal_type" => $row["proposal_type"],
"proposal_status" => $row["proposal_status"],
"order" => $row["order"],
"star" => $row["star"],
"del" => $row["del"]);       }
if (!$leads) {
$leads['status'] = 400;
$leads['message'] = 'Leads does not exists';
echo json_encode($leads);
}
  else {
            $leads['status'] = 200;
            echo json_encode($leads);
        }
    });
```

Kode sumber 29 Melihat data lead

```
$app->get("/performance", function() use ($app,
$db, $pdo) {
$data_user = $db->user()
->where("role_id = ?", "ae")
->where("del = ?", 0);

foreach ($data_user as $row) {
$data_project = $db->project()
->where("ae_id = ?", $row["id_user"])
->sum("project_value");
if($data_project == null) $data_project = "0";
$performa["data"]["performance_am"][] = array(
"name" => $row["fullname"],
"avatar" => $row["avatar"],
"total" => $data_project
);
}

$data_client_sector = $db->client_sector()
->where("del = ?", 0);
// $db_custom = new
PDO('mysql:dbhost=localhost;dbname=simple_crm','r
oot','');
//$pdo = PDO::getInstance();
foreach($data_client_sector as $row){
//native query
$id_sector = $row["id_sector"];
$stmt = $pdo->prepare("
SELECT SUM(project.project_value) as value
FROM Project
LEFT JOIN USER
ON project.il_id = user.id_user
LEFT JOIN client_sector
ON client_sector.id_sector = user.sector_id
WHERE user.sector_id = :id_sector
");

$stmt->bindParam(':id_sector', $id_sector);
```

```
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_ASSOC);

if($result["value"] == NULL) $result["value"] =
"0";
$performa["data"]["performance_segment"][] =
array(
"segment" => $row["sector"],
"total" => $result["value"]
);
}
$app->response()->header("Content-Type",
"application/json");
$performa["status"] = 200;
echo json_encode($performa);

});
```

Kode sumber 30 Melihat data performance

```
$app->get("/activities/:userId",
function($userId) use($app, $db) {
            $data = array('user_id' => $userId);
            $activities = $db->activities()-
>select("subject, create_date, due_date,
done_date")->where($data);
            $app->response()->header("Content-
Type", "application/json");
            $count = 0;

            foreach ($activities as $row) {
                $activities_data['data'] = array(
                    "agenda" => $row["subject"],
                    "createdata" =>
$row["create_date"],
                    "duedate" =>
$row["due_date"],
                    "donedate" =>
$row["done_date"]
                );
                $count++;
            }
            if($count == 0) {
                $activities_data['status'] = 400;
                $activities_data['message'] =
'data does not exists';
                echo
json_encode($activities_data);
            }
            else {
                $activities_data['status'] = 200;
```

Kode sumber 31 Melihat data tanggal aktifitas

```
public function getStructurOrganization($filter, $user_id) {
        if (is_array($filter))
            extract($filter);
        $str = '';

        if (!empty($s_number))
            $number = $s_number;

        if (!empty($number)) {
            $str .= " AND (LOWER(a.emp_number)
LIKE('%%$number%%') OR LOWER(a.emp_name)
LIKE('%%$number%%'))";
        }
        if (!empty($type_id) && ($type_id != 'all')) {
            $str .= " AND LOWER(v.`empcontract_emptype_id`) =
('$type_id')";
        }
        if (!empty($status_id) && ($status_id != 'all')) {
            $str .= " AND LOWER(p.`empstatus_empstat_id`) =
('$status_id')";
        }
        if (!empty($position_id) && ($position_id != 'all')) {
            $str .= " AND LOWER(t.`funcpos_funcpostype_id`) =
('$position_id')";
        }
        if (!empty($unit_id) && ($unit_id != 'all')) {
            $str .= " AND LOWER(r.`empunit_unit_id`) = ('$unit_id')";
        }
        $limit = '';
        if (!empty($display)) {
            $limit = "LIMIT $start, $display";
        }
        $query = $this->mSqlQueries['get_structur_organization'];
        $query = str_replace('--search--', $str, $query);
        $query = str_replace('--limit--', $limit, $query);
        $result = $this->Open(stripslashes($query), array($user_id));
        return $result;
    }
```

Kode sumber 32 Model Employee.Class.php

```
$sql["get_structur_organization"] = "
SELECT
  a.emp_id AS 'emp_id',
  a.emp_name AS 'emp_name',
  a.emp_number AS 'emp_number',
  a.emp_birth_place AS 'birth_place',
  a.`emp_birth_date` AS 'birth_date',
  c.strucpostype_id AS 'strucpos_id',
  c.strucpostype_parent_id AS 'strucpos_parentid',
  c.strucpostype_name AS 'strucpos_name',
  b.strucpos_start AS 'strucpos_start',
  b.strucpos_end AS 'strucpos_end',
  d.funcpos_funcpostype_id AS 'funcpos_id',
  d.funcpos_start AS 'funcpos_start',
  d.funcpos_end AS 'funcpos_end',
  e.`funcpostype_id` AS 'funcpos_id',
  e.`funcpostype_parent_id` AS 'funcpos_parentid',
  e.`funcpostype_name` AS 'funcpos_name'
FROM emp_employee a, emp_structural_position b
,ref_structural_position_type c , emp_functional_position d
,ref_functional_position_type e
WHERE
  b.strucpos_strucpostype_id = c.strucpostype_id AND
  b.`strucpos_emp_id` = a.`emp_id` AND
  d.`funcpos_funcpostype_id` = e.`funcpostype_id` AND
  d.`funcpos_emp_id` = a.`emp_id`
GROUP BY a.emp_name
";
```

Kode sumber 33 Model Employee.sql.php

```
<?php
class ViewEmployee extends HtmlResponse {
    function TemplateModule() {
        $this-
>SetTemplateBasedir(Configuration::Instance()-
>GetValue('application', 'docroot') . 'module/' .
GtfwDispt()->mModule . '/template');
        $this-
>SetTemplateFile('view_employee.html');
    }

    function ProcessRequest() {
        $ObjEmployee     =     GtfwDispt()->load-
>business('Employee', 'emp.employee');
        $ObjSetting      =     GtfwDispt()->load-
>business('Setting', 'core.setting');
        $user_id   =   Security::Authentication()-
>GetCurrentUser()->GetUserId();

        $msg       =       Messenger::Instance()-
>Receive(__FILE__);
        $filter_data    =    !empty($msg[0][0])    ?
$msg[0][0] : NULL;

        $urlSelect                                  =
!empty($msg[1]['url_select'])                      ?
$msg[1]['url_select']                              :
(!empty($msg[0]['url_select'])                     ?
$msg[0]['url_select'] : null);

        if      (!isset($_GET['display'])      ||
empty($filter_data)) {
            $page = 1;
            $start = 0;
            $display        =        $ObjSetting-
```

```
>getValue('view_per_page');
            $filter = compact('page', 'display',
'start');
        } elseif ($_GET['display']->Raw() != '')
{
            $page    =    (int)    $_GET['page']-
>SqlString()->Raw();
            $display  =  (int)  $_GET['display']-
>SqlString()->Raw();

            if ($page < 1)
                $page = 1;
            if ($display < 1)
                $display      =      $ObjSetting-
>getValue('view_per_page');
            $start = ($page - 1) * $display;

            $filter = compact('page', 'display',
'start');
            $filter += $filter_data;
        } else {
            $filter = $filter_data;
            $page = $filter['page'];
            $display = $filter['display'];
            $start = $filter['start'];
        }
$post_data = $_POST->AsArray();
        if (!empty($post_data)) {
            foreach   ($post_data   as   $key   =>
$value)
                $filter[$key] = $value;
            $filter['start'] = 0;
            $filter['page'] = $page = 1;
        }
```

```
        $total          =          $ObjEmployee-
>countEmployee($filter);
        $data           =          $ObjEmployee-
>getEmployee($filter);
        $user_active       =       $ObjEmployee-
>getUserActive();
        $data2          =          $ObjEmployee-
>getStructurOrganization();    //variabel    data2
menyimpan   nilai   pada   fungsi   sql   query   yang
menghasilkan data untuk orgchart. query ada pada
emp.employee

        Messenger::Instance()-
>Send('emp.structural.position',      'employee',
'view',          'html',          array($filter),
Messenger::UntilFetched);
        Messenger::Instance()-
>Send('emp.employee',  'employeeDetail',  'view',
'html', array($filter), Messenger::NextRequest);

        Messenger::Instance()-
>SendToComponent('emp.ref.employee.type',
'comboRefEmpType',  'view',  'html',  'type_id',
array(
            'dataId'                          =>
(!empty($filter['type_id']) ? $filter['type_id']
: null),
            'elmId' => 'type_id',
            'first' => 'all',
            'showAdd' => false,
            'name' => 'type_id',
            'style' => '',
            'script'          =>          ''),
Messenger::CurrentRequest);
```

```
        Messenger::Instance()-
>SendToComponent('emp.ref.employee.status',
'comboRefEmpStatus', 'view', 'html', 'status_id',
array(
            'dataId'                              =>
(!empty($filter['status_id'])                      ?
$filter['status_id'] : null),
            'elmId' => 'status_id',
            'first' => 'all',
            'showAdd' => false,
            'name' => 'status_id',
            'style' => '',
            'script'            =>            ''),
Messenger::CurrentRequest);

        Messenger::Instance()-
>SendToComponent('ref.functional.position.type',
'comboFunctionalPositionType',   'view',   'html',
'position_id', array(
            'dataId'                              =>
(!empty($filter['position_id'])                    ?
$filter['position_id'] : null),
            'elmId' => 'position_id',
            'first' => 'all',
            'showAdd' => false,
            'name' => 'position_id',
            'style' => '',
            'script'            =>            ''),
Messenger::CurrentRequest);

        Messenger::Instance()-
>SendToComponent('core.unit',         'comboUnit',
'view', 'html', 'unit_id', array(
            'dataId'                              =>
(!empty($filter['unit_id']) ? $filter['unit_id']
```

```
: null),
            'elmId' => 'unit_id',
            'first' => 'all',
            'showAdd' => false,
            'name' => 'unit_id',
            'style' => '',
            'script'            =>            ''),
Messenger::CurrentRequest);

        $url       =       Dispatcher::Instance()-
>GetUrl(Dispatcher::Instance()->mModule,
Dispatcher::Instance()->mSubModule,
Dispatcher::Instance()->mAction,
Dispatcher::Instance()->mType);
        Messenger::Instance()-
>SendToComponent('comp.paging', 'Paging', 'view',
'html',  'paging_top',  array($display,  $total,
$url, $page), Messenger::CurrentRequest);
        Messenger::Instance()-
>SendToComponent('comp.paging', 'Paging', 'view',
'html', 'paging_bottom', array($display, $total,
$url, $page), Messenger::CurrentRequest);

        return            compact('data','data2',
'msg_data', 'message', 'filter', 'urlSelect');
    }
function ParseTemplate($rdata = NULL) {
         $this->ButtonRendering();
        extract($rdata);


        $this->mrTemplate->addVar('search',
'URL',       GtfwDispt()->GetCurrentUrl()       .
'&display');
        if (!empty($filter)) {
```

```
            $this->mrTemplate->addVars('search',
$filter);
        }

        if (!empty($data) AND count($data) > 0) {
            $this->mrTemplate->addVar('data',
'IS_EMPTY', 'NO');
            $no = $filter['start'] + 1;
            foreach ($data as $val) {
                $val['no'] = $no;
                $val['url_detail'] = GtfwDispt()-
>GetUrl('emp.employee', 'detailEmployee', 'view',
'html') . '&no=' . $no . '&id=' . $val['id'];
                $val['url_curriculum_vitae']    =
GtfwDispt()->GetUrl('emp.employee',
'curriculumVitae', 'view', 'html') . '&no=' . $no
. '&id=' . $val['id'];

                $this->mrTemplate-
>clearTemplate('button_update');
                $this->mrTemplate-
>addVar('button_update',    'URL',    GtfwDispt()-
>GetUrl('emp.employee', 'updateEmployee', 'view',
'html') . '&id=' . $val['id']);

                $this->mrTemplate-
>clearTemplate('button_create_user');
                $this->mrTemplate-
>addVar('button_create_user', 'URL', GtfwDispt()-
>GetUrl('emp.employee.user',   'addEmployeeUser',
'view', 'html') . '&id=' . $val['id']);

                if       (in_array('create_user',
$actions)) {
                    if (!empty($val['user_id'])){
```

```
                        $this->mrTemplate-
>SetAttribute('button_create_user', 'visibility',
'hidden');
                    }else{
                        $max_user                =
Configuration::Instance()-
>GetValue('application', 'max_user');
                        if($user_active          >=
$max_user){
```

Kode sumber 34 Controller ViewEmployee.Class.php

```
<script type="text/javascript">
(function($) {
    $.fn.orgChart = function(options) {
        var opts = $.extend({}, $.fn.orgChart.defaults, options);
        return new OrgChart($(this), opts);
    }

    $.fn.orgChart.defaults = {
        data: [{id:1, name:'Root', parent: 0, posisi:''}],
        showControls: false,
        allowEdit: false,
        onAddNode: null,
        onDeleteNode: null,
        onClickNode: null,
        newNodeText: 'Add Child'
    };

    function OrgChart($container, opts){
        var data = opts.data;
        var nodes = {};
        var rootNodes = [];
        this.opts = opts;
        this.$container = $container;
        var self = this;

        this.draw = function(){
            $container.empty().append(rootNodes[0].render(opts));
            $container.find('.node').click(function(){
                if(self.opts.onClickNode !== null){
                    self.opts.onClickNode(nodes[$(this).attr('node-
id')]);
                }
            });

            if(opts.allowEdit){
                $container.find('.node h2').click(function(e){
                    var thisId = $(this).parent().attr('node-id');
                    self.startEdit(thisId);
                    e.stopPropagation();
                });
            }

            // add "add button" listener
            $container.find('.org-add-button').click(function(e){
                var thisId = $(this).parent().attr('node-id');

                if(self.opts.onAddNode !== null){
                    self.opts.onAddNode(nodes[thisId]);
                }
                else{
                    self.newNode(thisId);
                }
                e.stopPropagation();
            });

            $container.find('.org-del-button').click(function(e){
```

```
                var thisId = $(this).parent().attr('node-id');

                if(self.opts.onDeleteNode !== null){
                    self.opts.onDeleteNode(nodes[thisId]);
                }
                else{
                    self.deleteNode(thisId);
                }
                e.stopPropagation();
            });
        }

        this.startEdit = function(id){
            var inputElement = $('<input class="org-input" type="text"
value="'+nodes[id].data.name+'"/>');
            $container.find('div[node-id='+id+']
h2').replaceWith(inputElement);
            var commitChange = function(){
                var h2Element = $('<h2>'+nodes[id].data.name+'</h2>');
                if(opts.allowEdit){
                    h2Element.click(function(){
                        self.startEdit(id);
                    })
                }
                inputElement.replaceWith(h2Element);
            }
            inputElement.focus();
            inputElement.keyup(function(event){
                if(event.which == 13){
                    commitChange();
                }
                else{
                    nodes[id].data.name = inputElement.val();
                }
            });
            inputElement.blur(function(event){
                commitChange();
            })
        }

        this.newNode = function(parentId){
            var nextId = Object.keys(nodes).length;
            while(nextId in nodes){
                nextId++;
            }

            self.addNode({id: nextId, name: '', parent: parentId});
        }

        this.addNode = function(data){
            var newNode = new Node(data);
            nodes[data.id] = newNode;
            nodes[data.parent].addChild(newNode);

            self.draw();
            self.startEdit(data.id);
```

```
        }

        this.deleteNode = function(id){
            for(var i=0;i<nodes[id].children.length;i++){
                self.deleteNode(nodes[id].children[i].data.id);
            }
            nodes[nodes[id].data.parent].removeChild(id);
            delete nodes[id];
            self.draw();
        }

        this.getData = function(){
            var outData = [];
            for(var i in nodes){
                outData.push(nodes[i].data);
            }
            return outData;
        }

        // constructor
        for(var i in data){
            var node = new Node(data[i]);
            nodes[data[i].id] = node;
        }

        // generate parent child tree
        for(var i in nodes){
            if(nodes[i].data.parent == 0){
                rootNodes.push(nodes[i]);
            }
            else{
                nodes[nodes[i].data.parent].addChild(nodes[i]);
            }
        }

        // draw org chart
        $container.addClass('orgChart');
        self.draw();
    }

    function Node(data){
        this.data = data;
        this.children = [];
        var self = this;

        this.addChild = function(childNode){
            this.children.push(childNode);
        }

        this.removeChild = function(id){
            for(var i=0;i<self.children.length;i++){
                if(self.children[i].data.id == id){
                    self.children.splice(i,1);
                    return;
                }
            }
```

```
        }

        this.render = function(opts){
            var childLength = self.children.length,
                mainTable;

            mainTable = "<table cellpadding='0' cellspacing='0'
border='0'>";
            var nodeColspan = childLength>0?2*childLength:2;
            mainTable += "<tr><td
colspan='"+nodeColspan+"'>"+self.formatNode(opts)+"</td></tr>";

            if(childLength > 0){
                var downLineTable = "<table cellpadding='0'
cellspacing='0' border='0'><tr class='lines x'><td class='line left
half'></td><td class='line right half'></td></table>";
                mainTable += "<tr class='lines'><td
colspan='"+childLength*2+"'>"+downLineTable+'</td></tr>';

                var linesCols = '';
                for(var i=0;i<childLength;i++){
                    if(childLength==1){
                        linesCols += "<td class='line left
half'></td>";    // keep vertical lines aligned if there's only 1
child
                    }
                    else if(i==0){
                        linesCols += "<td class='line left'></td>";
// the first cell doesn't have a line in the top
                    }
                    else{
                        linesCols += "<td class='line left
top'></td>";
                    }

                    if(childLength==1){
                        linesCols += "<td class='line right
half'></td>";
                    }
                    else if(i==childLength-1){
                        linesCols += "<td class='line right'></td>";
                    }
                    else{
                        linesCols += "<td class='line right
top'></td>";
                    }
                }
                mainTable += "<tr class='lines v'>"+linesCols+"</tr>";

                mainTable += "<tr>";
                for(var i in self.children){
                    mainTable += "<td
colspan='2'>"+self.children[i].render(opts)+"</td>";
                }
                mainTable += "</tr>";
            }
```

```
            mainTable += '</table>';
            return mainTable;
        }

        this.formatNode = function(opts){
            var nameString = '',
                descString = '';
            var detailid = "detail"+this.data.id;
            if(typeof data.name !== 'undefined'){
                nameString = '<h2>'+self.data.name+'</h2>';
            }
            if(typeof data.detail !== 'undefined'){
                detailString = self.data.detail;
            }
            if(typeof data.posisi !== 'undefined'){
                descString = '<p>'+self.data.posisi+'</p>';
            }
            if(typeof data.name !== 'undefined'){
              var fotoString = '<p class="centered"><img
src="'+self.data.avatar+'avatar-default.png" class="img-circle"
width="60"></p>';
            }

            if(opts.showControls){
                //var buttonsHtml = "<div class='org-add-
button'>"+opts.newNodeText+"</div><div class='org-del-button'></div>";
                var buttonsHtml = "<a href='#"+detailid+"'
id='addcl_btn' data-toggle='modal'><p class='centered'><button>See
Details</button></p></a>";
            }
            else{
                buttonsHtml = '';
            }
            //return "<div class='node' node-
id='"+this.data.id+"'>"+fotoString+descString+buttonsHtml+"</div><div
id='"+detailid+"' class='detail'
style='display:none;'>"+detailString+"</div>";
            return "<div class='node' node-
id='"+this.data.id+"'>"+fotoString+descString+buttonsHtml+"</div><div
class='modal fade' id='"+detailid+"' tabindex='-1' role='dialog' aria-
labelledby='myModalLabel' aria-hidden='true'><div class='modal-
content'><div class='modal-header'><button type='button' class='close'
data-dismiss='modal' aria-hidden='true'>&times;</button><h4
class='modal-title' id='myModalLabel'>"+descString+"</h4></div><div
class='modal-body'><div class='table-responsive'><table class='table
table-striped mb30'><thead><tr><th><center>No</center></th><th
colspan='2'><center>Members</center></th></tr></thead><tbody
id='content-detail'>"+detailString+"</tbody></table></div></div><div
class='modal-footer'><div class='loading-state pull-left'
style='display:none;'><p><img src='<?php echo
base_url('assets/images/loaders/loader8.gif') ?>'/> Please wait . .
.</p></div></div></div></div>";
        }
    }

})(jQuery);
```

```
</script>

<script type="text/javascript">
var total = 10000;
var testData = new Object();
var count = 1;
for (var i = 0; i < total; i++) {
    if(document.getElementById("strucpos_id"+count)){
        var avatar = document.getElementById("avatar").value;
        var strucpos_parentid =
document.getElementById("strucpos_parentid"+count).value;
        var strucpos_name =
document.getElementById("strucpos_name"+count).value;
        var strucpos_id =
document.getElementById("strucpos_id"+count).value;
        var str = document.getElementById("emp_name"+count).value;
        var emp_name = str.slice(0,22);
        var count2 = 1 ;
        var count3 = 1 ;
        var total2 = 99999;
        var content = "";
        var detailid = "detail"+strucpos_id;
        for (var j = 0; j < total2; j++) {
            if(document.getElementById("strucpos_id"+count2)){
                var node =
document.getElementById("strucpos_id"+count2).value;
                if(node == strucpos_id){
                    var image =
avatar+document.getElementById("emp_id"+count2).value+".jpg";
                    var imagedefault =
"this.onerror=null;this.src='files/employee/avatar-default.png'";
                    content = content+'<tr><td
rowspan="2">'+count3+'</td><td rowspan="2"><img src="'+image+'"
width="60" onError="'+imagedefault+'";"><td align="left"><p
align="left" style="font-size:150%;margin-
bottom:0px;padding:0px;">'+document.getElementById("emp_name"+count2).
value+'</p></td></tr><tr><td><p align="left" style="font-
size:110%;margin-bottom:0px;padding:0px;font-
style:italic;">'+document.getElementById("funcpos_name"+count2).value+
'</p></td></tr>';
                    count3++;
                }
                count2 = count2 + 1;
            }
            else{
                total2 = 0;
            }
        }
        testData[i] = ({
            avatar : avatar,
            id: strucpos_id ,
            name: emp_name,
            parent: strucpos_parentid,
            posisi: strucpos_name,
            detail : content,
            detailid : detailid,
```

```
            });
            count=count + 1;
        }
        else{
            total = 0;
        }
}
function viewdetail(id){
        var x = event.clientX;     // Get the horizontal coordinate
        var y = event.clientY;     // Get the vertical coordinate
        x = x + 80;
        y = y- 120;
        var coor = "X coords: " + x + ", Y coords: " + y;
        var detailid = "detail"+id;
        var status = document.getElementById(detailid).style.display;
        if(status != 'none'){
            document.getElementById(detailid).style.display = 'none';
        }
        else{
            document.getElementById(detailid).style.position = 'fixed';
            document.getElementById(detailid).style.display = 'inline-
block';
            document.getElementById(detailid).style.left = x+'px';
            document.getElementById(detailid).style.top = y+'px';
        }
        //alert(coor);
}

$(function(){
        org_chart = $('#orgChart').orgChart({
            data: testData,
            showControls: true,
            allowEdit: false,
            onClickNode: function(node){
                //alert("I am an alert box!");
                //log('Clicked node '+node.data.id);
            }

        });
});
function keydown() {
        var total = 1000;
        var count = 1;
        for (var i = 0; i < total; i++) {
            if(document.getElementById("position_id"+count)){
                var id =
document.getElementById("position_id"+count).value;
                var detailid = "detail"+id;
                document.getElementById(detailid).style.display = 'none';
                count=count +1;
            }
            else{
                total = 0;
            }
        }
}
```

```
$(document).ready(function()
{
    $("#orgChart").mouseup(function(e)
    {
        var total = 1000;
        var count = 1;
        for (var i = 0; i < total; i++) {
            if(document.getElementById("position_id"+count)){
                var id =
document.getElementById("position_id"+count).value;
                var detailid = "#detail"+id;
                var subject = $(detailid);
                if(e.target.id != subject.attr('id'))
                {
                    subject.fadeOut();
                }
                count=count +1;
            }
            else{
                total = 0;
            }
        }
    });

    $(window).scroll(function() {
        var total = 1000;
        var count = 1;
        for (var i = 0; i < total; i++) {
            if(document.getElementById("position_id"+count)){
                var id =
document.getElementById("position_id"+count).value;
                var detailid = "detail"+id;
                document.getElementById(detailid).style.display =
'none';
                count=count +1;
            }
            else{
                total = 0;
            }
        }
    });
});
</script>
```

Kode sumber 35 template view_employee.html

BIODATA PENULIS

Nama	:	Alifa Ridho Musthofa
TTL	:	Klaten, 8 November 1993
Jenis Kelamin	:	Laki-laki
Alamat	:	RT02/01 Puluhan, Trucuk, Klaten
No. Telepon	:	085728640629
Email	:	alif.sip@gmail.com

Nama	:	Alief Yoga Priyanto
TTL	:	Surakarta, 7 Mei 1994
Jenis Kelamin	:	Laki-laki
Alamat	:	Jalan Jayawijaya VI/12, Mojosongo, Solo
No. Telepon	:	083866699908
Email	:	aliefyp@gmail.com