



KERJA PRAKTIK - KI141330

**RANCANG BANGUN APLIKASI MEME FLORIST
DENGAN GOOGLE APP ENGINE BERBASIS
PERANGKAT MOBILE ANDROID UNTUK PEMESANAN
BUNGA ONLINE**

Nabilla Sabbaha Audria Permatasari
NRP 5111 100 015

Ardhiansyah Baskara
NRP 5111 100 088

Dosen Pembimbing
Wijayanti Nurul Khotimah, S.Kom, M.Sc.
NIP. 198603122012122004

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014



KERJA PRAKTIK - KI141330

**RANCANG BANGUN APLIKASI MEME FLORIST
DENGAN GOOGLE APP ENGINE BERBASIS
PERANGKAT MOBILE ANDROID UNTUK PEMESANAN
BUNGA ONLINE**

Nabilla Sabbaha Audria Permatasari
NRP 5111 100 015

Ardhiansyah Baskara
NRP 5111 100 088

Dosen Pembimbing
Wijayanti Nurul Khotimah, S.Kom, M.Sc.
NIP. 198603122012122004

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

LEMBAR PENGESAHAN I

Judul : *Rancang Bangun Aplikasi Meme Florist dengan Google App Engine Berbasis Perangkat Mobile Android untuk Pemesanan Bunga Online*

Lokasi : PT. Tunas Sinergi Sejahtera

Periode : 23 Juni 2014 – 23 Juli 2014

Jakarta, 18 Agustus 2014

Mengetahui,

Pembimbing Lapangan



Oon Arfiandwi

CTO 7Langit

LEMBAR PENGESAHAN II

Judul : *Rancang Bangun Aplikasi Meme Florist dengan Google App Engine Berbasis Perangkat Mobile Android untuk Pemesanan Bunga Online*

Lokasi : PT. Tunas Sinergi Sejahtera

Periode : 23 Juni 2014 – 24 Juli 2014

Surabaya, 21 Agustus 2014

Dosen Pembimbing

Wijayanti Nurul Khotimah, S.Kom, M.Sc.

NIP. 198603122012122004

RANCANG BANGUN APLIKASI MEME FLORIST DENGAN GOOGLE APP ENGINE BERBASIS PERANGKAT MOBILE ANDROID UNTUK PEMESANAN BUNGA ONLINE

ABSTRAK

Semakin berkembangnya pasar aplikasi mobile saat ini, mendorong manusia untuk dapat menangani segala kegiatan dalam hidupnya dengan teknologi. Alasannya adalah agar segala kegiatan yang dijalani lebih cepat, efisien dan dapat dilakukan dimana saja dan kapan saja. Satu-satunya yang dapat dilakukan manusia untuk mewujudkan hal-hal tersebut adalah dengan memaksimalkan teknologi. Salah satunya melalui perangkat mobile Android yang sekarang sedang marak digunakan.

Kegiatan yang dapat dilakukan dengan memaksimalkan perangkat mobile android salah satunya adalah melakukan pemesanan bunga online yang perancangan dan implementasinya telah kami kerjakan. Aplikasi yang databasenya memanfaatkan Google Cloud Engine ini, membuat pengguna tidak perlu repot untuk datang ke toko bunga untuk sekedar melihat katalog bunga dan memesan bunga, namun pengguna kini dapat melihat katalog dan memesan bunga dimanapun dan kapanpun pengguna berada melalui smartphone androidnya. Selain itu, pengguna juga bisa mendapatkan notifikasi mengenai status pengiriman bunga yang telah dipesan dan notifikasi promosi mengenai produk bunga

yang dimiliki Meme Florist. Notifikasi yang digunakan dalam aplikasi ini memanfaatkan Google Cloud Messaging untuk mengirimkan notifikasi secara broadcast maupun secara personal untuk penggunanya.

Selain pengguna, administrator Meme Florist juga dapat merasakan kemudahan dalam mengelola semua informasi yang berada di aplikasi yang dimiliki pengguna melalui web administrator. Administrator dapat melakukan pengiriman konfirmasi pemesanan bunga, konfirmasi reset password, dan konfirmasi aktivasi akun baru. Untuk keperluan pengiriman notifikasi, Administrator dapat mengirimkan notifikasi secara broadcast bagi seluruh penggunanya dan secara personal bagi pengguna tertentu.

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah SubhanahuWata'alla yang telah melimpahkan rahmat dan hidayah-Nya hingga penulis dapat menyelesaikan Laporan Kerja Praktik yang berjudul “RANCANG BANGUN APLIKASI MEME FLORIST DENGAN GOOGLE APP ENGINE BERBASIS PERANGKAT MOBILE ANDROID UNTUK PEMESANAN BUNGA ONLINE” dengan baik.

Dalam pelaksanaan dan pembuatan Laporan Kerja Praktik ini tentunya penulis banyak mendapatkan banyak bantuan dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih kepada:

1. Kedua orang tua penulis yang telah memberikan banyak dukungan moral, spiritual dan material selama kerja praktik berlangsung.
2. Ibu Wijayanti Nurul Khotimah, S.Kom, M.Sc., selaku dosen pembimbing kerja praktik yang telah memberikan kepercayaan, dukungan, dan bimbingan kepada penulis.
3. Ibu Nanik Suciati, S.Kom., M.Kom., Dr.Eng. selaku ketua jurusan Teknik Informatika ITS, Ibu Handayani Tjandrasa, M.Sc, Ph.D., Prof.Ir. selaku dosen wali penulis, dan Bapak Radityo Anggoro, S.Kom, M.Sc., Dr.Eng. selaku koordinator KP.
4. Ibu Titi Rusdi, selaku CEO PT.Tunas sinergi Sejahtera (7Langit) dan Bapak Oon Arfiandwi selaku CTO PT.Tunas sinergi Sejahtera (7Langit) yang telah memberikan kesempatan untuk menimba ilmu kerja praktik, serta teman-teman 7Langit yang telah memberikan ilmu yang berharga mengenai *software process development* pada penulis.
5. Teman-teman penulis yang telah turut membantu selesainya Laporan Kerja Praktik.

Surabaya, 20 Agustus 2014

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN I	v
LEMBAR PENGESAHAN II	vii
ABSTRAK	xi
KATA PENGANTAR.....	xiii
DAFTAR ISI	xv
DAFTAR GAMBAR	xxi
DAFTAR TABEL	xxxvi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan dan Manfaat.....	3
1.4 Waktu Pelaksanaan.....	4
1.5 Sistematika Penulisan.....	4
BAB II PROFIL PERUSAHAAN	5
2.1 Sejarah Perusahaan.....	5
2.2 Logo Perusahaan	6
2.3 Visi dan Misi Perusahaan	6
BAB III TINJAUAN PUSTAKA.....	7
3.1 Android.....	7
3.2 Keunggulan Android	8

3.3	Android SDK.....	9
3.4	Bahasa Pemrograman Java dan XML	10
3.5	Google Cloud Messaging (GCM)	10
3.6	Google Maps	12
3.7	Google App Engine	12
3.8	Google Cloud Endpoints	13
3.9	Java Datastore API.....	14
3.10	Java Data Objects (JDO) with AppEngine.....	14
3.11	Blobstore Java API.....	15
3.12	Apache Velocity	15
	BAB IV ANALISIS DAN PERANCANGAN SISTEM	17
4.1.	Analisis dan Perancangan Aplikasi <i>Mobile</i>	17
4.1.1.	Analisis Kebutuhan	17
4.1.2.	Definisi Umum Aplikasi.....	21
4.1.3.	Perancangan Fungsionalitas Aplikasi.....	23
4.2.	Analisis dan Perancangan Aplikasi <i>Web</i>	52
4.2.1	Analisis Kebutuhan	52
4.2.2	Definisi Umum Aplikasi.....	54
4.2.3	Perancangan Fungsionalitas Aplikasi.....	55
	BAB V IMPLEMENTASI.....	95

5.1	Implementasi Aplikasi Meme Florist	95
5.1.1.	Melihat kategori bunga dan detil bunga	95
5.1.2.	Melihat informasi tentang profil dan kontak Meme Florist	98
5.1.3.	Mengelola keranjang belanja.....	103
5.1.4.	Memperbarui profil pengguna.....	104
5.1.5.	Melihat histori pemesanan dan detail histori pemesanan	106
5.1.6.	Menampilkan histori notifikasi.....	109
5.1.7.	Melakukan registrasi	110
5.1.8.	Melakukan pemesanan bunga.....	111
5.1.9.	Melakukan permintaan <i>reset password</i>	111
5.1.10.	Menerima <i>push notification</i> dari Meme Florist.	112
5.1.11.	Melakukan pengaturan penerimaan <i>push notification</i>	113
5.2	Implementasi Website Administrator Meme Florist..	115
5.2.1.	<i>Login</i> Administrator Meme Florist.....	115
5.2.2.	Menampilkan Halaman Beranda Meme Florist.	118
5.2.3.	Melihat Daftar Katalog Bunga	120
5.2.4.	Menambah Data Katalog Bunga	122
5.2.5.	Mengubah Data Katalog Bunga	127

5.2.6.	Menghapus Data Katalog Bunga.....	131
5.2.7.	Melihat Daftar Pemesanan Pelanggan	133
5.2.8.	Melakukan Konfirmasi Pemesanan Bunga.....	135
5.2.9.	Mengirim Pemberitahuan kepada Pelanggan	137
5.2.10.	Mengubah Profil dan Kontak Toko Meme Florist	
	145	
5.2.11.	Melakukan Aktivasi Akun Pelanggan	149
5.2.12.	Mengubah <i>Password</i> Akun Pelanggan.....	153
5.2.13.	API (<i>Application Programming Interface</i>) Florist Endpoint	
	156	
BAB VI UJI COBA DAN EVALUASI		179
6.1	Aplikasi Perangkat <i>Mobile</i> Meme Florist.....	179
6.1.1	Lingkungan Uji Coba	179
6.1.2	Menampilkan Hasil Uji Coba	179
6.2	Aplikasi <i>Web</i> Administrator Meme Florist	195
6.2.1	Lingkungan Uji Coba	195
6.2.2	Menampilkan Hasil Uji Coba	196
BAB VII KESIMPULAN DAN SARAN		215
7.1.	Kesimpulan.....	215
7.2.	Saran.....	216
DAFTAR PUSTAKA.....		217



Rancang Bangun Aplikasi Meme Florist dengan Google App Engine Berbasis Perangkat Mobile Android untuk Pemesanan Bunga Online
Kantor PT. Tunas Sinergi Sejahtera (7Langit) Jakarta Jalan
Moch.Yamin No.1 Jakarta Pusat - DKI Jakarta
23 Juni 2014 – 23 Juli 2014

LAMPIRAN 219

DAFTAR GAMBAR

Gambar 2.1 Logo 7Langit	6
Gambar 3.1 Arsitektur Google Cloud Endpoints	13
Gambar 3.2 Skema Cara Kerja Apache Velocity	16
Gambar 4.1 Kasus Penggunaan Aplikasi <i>Mobile Meme Florist</i> .	23
Gambar 4.2 Diagram Aktivitas untuk Kasus Melihat Katalog Bunga	29
Gambar 4.3 Diagram Aktivitas untuk Melihat Informasi tentang Meme Florist	30
Gambar 4. 4 Diagram Aktivitas untuk Mengelola Keranjang Belanja.....	Error! Bookmark not defined.
Gambar 4.5 Diagram Aktivitas untuk Kasus Mengelola Profil Pengguna	32
Gambar 4.6 Diagram Aktivitas untuk Kasus Melihat Histori Pemesanan.....	33
Gambar 4.7 Diagram Aktivitas untuk Kasus Melihat Histori Notifikasi.....	33
Gambar 4.8 Diagram Aktivitas untuk Kasus Registrasi.....	34
Gambar 4.9 Diagram Aktivitas untuk Kasus Meminta <i>Reset Password</i>	34
Gambar 4.10 Diagram Aktivitas untuk Kasus Memesan Bunga.	35

Gambar 4.11 Diagram Aktivitas untuk Kasus Menerima <i>Push Notification</i>	35
Gambar 4.12 Diagram Aktivitas untuk Kasus Mengatur Penerimaan Notifikasi	36
Gambar 4.13 Rancangan antarmuka <i>splashscreen</i>	Error! Bookmark not defined.
Gambar 4.14 Rancangan antarmuka <i>landing menu</i> pengguna tidak <i>login</i>	Error! Bookmark not defined.
Gambar 4. 16 Rancangan antarmuka katalog pada detail bunga	Error! Bookmark not defined.
Gambar 4. 15 Rancangan antarmuka katalog pada kategori bunga	Error! Bookmark not defined.
Gambar 4.17 Rancangan antarmuka halaman informasi tentang Meme Florist	Error! Bookmark not defined.
Gambar 4.18 Rancangan antarmuka halaman kontak Meme Florist	Error! Bookmark not defined.
Gambar 4.20 Rancangan antarmuka cara pembayaran	Error! Bookmark not defined.
Gambar 4.19 Rancangan antarmuka lokasi kami	Error! Bookmark not defined.
Gambar 4.21 Rancangan antarmuka <i>landing menu</i> untuk pengguna yang telah <i>login</i>	Error! Bookmark not defined.
Gambar 4. 22 Rancangan antarmuka halaman kategori bunga	Error! Bookmark not defined.

Gambar 4. 24 Rancangan antarmuka halaman masukkan ke keranjang **Error! Bookmark not defined.**

Gambar 4. 23 Rancangan antarmuka halaman detail bunga **Error! Bookmark not defined.**

Gambar 4.25 Rancangan antarmuka halaman akun saya**Error! Bookmark not defined.**

Gambar 4.26 Rancangan antarmuka halaman edit akun saya **Error! Bookmark not defined.**

Gambar 4. 28 Rancangan antarmuka halaman histori pemesanan **Error! Bookmark not defined.**

Gambar 4. 27 Rancangan antarmuka halaman detail histori pemesanan **Error! Bookmark not defined.**

Gambar 4. 30 Rancangan antarmuka halaman keranjang belanja **Error! Bookmark not defined.**

Gambar 4.29 Rancangan antarmuka halaman histori notifikasi **Error! Bookmark not defined.**

Gambar 4.33 Rancangan antarmuka halaman *forgot password* **Error! Bookmark not defined.**

Gambar 4.31 Rancangan antarmuka halaman *login*Error! Bookmark not defined.

Gambar 4.32 Rancangan antarmuka halaman registrasi**Error! Bookmark not defined.**

Gambar 4.34 Rancangan antarmuka halaman pesan bunga .**Error! Bookmark not defined.**

Gambar 4.35 Kasus Penggunaan Aplikasi Mobile Web Meme Florist	55
Gambar 4.36 Diagram Aktivitas untuk Kasus Melihat Daftar Katalog Bunga.....	60
Gambar 4.37 Diagram Aktivitas untuk Kasus Menambah Data Katalog Bunga.....	61
Gambar 4.38 Diagram Aktivitas untuk Kasus Mengubah Data Katalog Bunga.....	62
Gambar 4.39 Diagram Aktivitas untuk Kasus Menghapus Data Katalog Bunga.....	63
Gambar 4.40 Diagram Aktivitas untuk Kasus Melihat Daftar Pemesanan Pelanggan	64
Gambar 4.41 Diagram Aktivitas untuk Kasus Melakukan Konfirmasi Pemesanan Bunga	65
Gambar 4.42 Diagram Aktivitas untuk Kasus Mengirim Pemberitahuan kepada Pelanggan	66
Gambar 4.43 Diagram Aktivitas untuk Kasus Mengubah Profil dan Kontak Toko Meme Florist	67
Gambar 4.44 Diagram Aktivitas untuk Kasus Melakukan Aktivasi Akun Pelanggan	68
Gambar 4.45 Diagram Aktivitas untuk Kasus Mengubah <i>Password</i> Akun Pelanggan	69
Gambar 4.46 Rancangan antarmuka halaman <i>login</i> administrator	78

Gambar 4.47 Rancangan antarmuka halaman daftar katalog bunga	79
Gambar 4.48 Rancangan antarmuka <i>item</i> katalog bunga	79
Gambar 4.49 Rancangan antarmuka menambah data katalog bunga	81
Gambar 4.50 Rancangan antarmuka mengubah data katalog bunga	82
Gambar 4.51 Rancangan antarmuka halaman detail katalog	83
Gambar 4.52 Rancangan antarmuka menampilkan daftar pemesanan pelanggan.....	84
Gambar 4.53 Rancangan antarmuka konfirmasi pemesanan pelanggan	85
Gambar 4.54 Rancangan antarmuka mengirim pemberitahuan kepada pelanggan	86
Gambar 4.55 Rancangan antarmuka menampilkan pengaturan profil dan kontak Meme Florist.....	87
Gambar 4.56 Rancangan antarmuka pengubahan profil dan kontak Meme Florist	90
Gambar 4.57 Rancangan antarmuka halaman notifikasi sukses aktivasi akun pelanggan	91
Gambar 4.58 Rancangan antarmuka halaman notifikasi gagal aktivasi akun pelanggan	91
Gambar 4.59 Rancangan antarmuka halaman <i>reset password</i> akun pelanggan	93

Gambar 4.60 Rancangan antarmuka notifikasi sukses <i>reset password</i> akun pelanggan	94
Gambar 4.61 Rancangan antarmuka notifikasi gagal <i>reset password</i> akun pelanggan	94
Gambar 5.1 Kode program fragment_category_katalog.xml	95
Gambar 5.2 Kode fungsi doInBackground asynctask KatalogFragment	96
Gambar 5.3 Kode fungsi onPostExecute asynctask KatalogFragment	97
Gambar 5.4 Kode program activity_page_bunga.xml	97
Gambar 5.5 Kode fungsi doInBackground asynctask PagerBungaActivity	99
Gambar 5.6 Kode fungsi onPostExecute asynctask PagerBungaActivity	99
Gambar 5.7 Kode fungsi doInBackground asynctask kelas Company	100
Gambar 5.8 Kode program activity_hubungi_kami.xml	100
Gambar 5.9 Kode fungsi doInBackground asynctask HubungiKamiFragment	100
Gambar 5.10 Kode program cara_pembayaran.xml	101
Gambar 5.11 Kode fungsi onCreate	102
Gambar 5.12 Kode program activity_location_maps.xml	102

Gambar 5.13 Kode fungsi initializeMap	102
Gambar 5.14 Kode fungsi onClick pada AlertDialog	103
Gambar 5.15 Kode program ActivityKeranjangBelanja.xml	104
Gambar 5.16 Kode fungsi onCreateView	105
Gambar 5.17 Kode fungsi hitungTotal.....	105
Gambar 5.18 Kode fungsi onCreateView	106
Gambar 5.19 Kode fungsi doInBackground asynctask EditAkunSaya.....	107
Gambar 5.20 Kode program fragment_histori_pemesanan.xml	107
Gambar 5.21 Kode fungsi doInBackground asynctask HistoriPemesananFragment.....	108
Gambar 5.22 Kode program activity_histori_pemesanan_listview.xml	108
Gambar 5.23 Kode fungsi onCreate	109
Gambar 5.24 Kode program fragmetn_push_notif.xml	109
Gambar 5.25 Kode fungsi doInBackground asynctask HistoriNotifikasiFragment.....	110
Gambar 5.26 Kode fungsi doInBackground asynctask RegisterActivity	110
Gambar 5.27 Kode fungsi doInBackground asynctask OrderActivity	111

Gambar 5.28 Kode fungsi doInBackground asynctask ForgotPassword.....	112
Gambar 5.29 Kode fungsi registerAtGCM	113
Gambar 5.30 Kode fungsi showNotification.....	114
Gambar 5.31 Kode fungsi registerGCMClient.....	114
Gambar 5.32 Kode tampilan login.vm	115
Gambar 5.33 Kode fungsi doGet LoginServlet.....	117
Gambar 5.34 Kode fungsi doPost LoginServlet.....	118
Gambar 5.35 Kode tampilan dashboard.vm.....	118
Gambar 5.36 Kode fungsi doGet DashboardServlet	120
Gambar 5.37 Kode fungsi doPost DashboardServlet	120
Gambar 5.38 Kode tampilan catalog.vm (<i>view</i>).....	121
Gambar 5.39 Kode fungsi doGet CatalogServlet (<i>view</i>)	123
Gambar 5.40 Kode tampilan catalog.vm (<i>add</i>)	124
Gambar 5.41 Kode fungsi doGet CatalogAddServlet	126
Gambar 5.42 Kode fungsi doPost CatalogAddServlet	127
Gambar 5.43 Kode tampilan catalog.vm (<i>update</i>)	128
Gambar 5.44 Kode fungsi doGet CatalogUpdateServlet	130
Gambar 5.45 Kode fungsi doPost CatalogUpdateServlet	131
Gambar 5.46 Lanjutan kode fungsi doPost CatalogUpdateServlet	132

Gambar 5.47 Kode tampilan catalog.vm (<i>delete</i>).....	132
Gambar 5.48 Kode fungsi doGet CatalogServlet (<i>delete</i>).....	134
Gambar 5.49 Kode tampilan order.vm (<i>view</i>)	138
Gambar 5.50 Kode fungsi doGet OrderServlet	139
Gambar 5.51 Kode tampilan order.vm (<i>confirm</i>).....	139
Gambar 5.52 Lanjutan kode tampilan order.vm (<i>confirm</i>).....	140
Gambar 5.53 Kode fungsi doGet ConfirmServlet	140
Gambar 5.54 Kode fungsi doPost ConfirmServlet.....	141
Gambar 5.55 Lanjutan kode fungsi doPost ConfirmServlet.....	142
Gambar 5.56 Kode tampilan notification.vm	143
Gambar 5.57 Kode fungsi doGet SendNotificationServlet	144
Gambar 5.58 Kode fungsi doPost SendNotificationServlet	145
Gambar 5.59 Lanjutan kode fungsi doPost SendNotificationServlet	147
Gambar 5.60 Kode tampilan settings.vm (<i>update</i>).....	148
Gambar 5.61 Lanjutan kode tampilan settings.vm (<i>update</i>)	149
Gambar 5.62 Kode fungsi doGet SettingsServlet.....	150
Gambar 5.63 Kode fungsi doPost SettingsServlet	152
Gambar 5.64 Lanjutan kode fungsi doPost SettingsServlet	152
Gambar 5.65 Kode tampilan activation.vm.....	153
Gambar 5.66 Kode fungsi doGet ActivationServlet.....	154

Gambar 5.67 Kode tampilan reset.vm.....	154
Gambar 5.68 Kode fungsi doGet ResetPasswordServlet	155
Gambar 5.69 Kode fungsi doPost ResetPasswordServlet	157
Gambar 5.70 Kode fungsi getCatalogs.....	158
Gambar 5.71 Kode fungsi getCategories.....	158
Gambar 5.72 Kode fungsi getCompanyContacts	159
Gambar 5.73 Kode fungsi getUsers	160
Gambar 5.74 Kode fungsi getGCMBroadcastDevices.....	160
Gambar 5.75 Kode fungsi getGCMPersonalDevices	161
Gambar 5.76 Kode fungsi getOrders.....	161
Gambar 5.77 Kode fungsi getCompanyProfile	162
Gambar 5.78 Kode fungsi loginCompany	163
Gambar 5.79 Kode fungsi getCatalogsByCategory	163
Gambar 5.80 Kode fungsi historyOrders.....	164
Gambar 5.81 Kode fungsi historyNotifications.....	164
Gambar 5.82 Kode fungsi historyShipments.....	165
Gambar 5.83 Kode fungsi getCatalogById	166
Gambar 5.84 Kode fungsi getShipmentById	166
Gambar 5.85 Kode fungsi getUserById	167
Gambar 5.86 Kode fungsi getCategoryById.....	167

Gambar 5.87 Kode fungsi getOrderBy Id	168
Gambar 5.88 Kode fungsi addCatalog	168
Gambar 5.89 Kode fungsi addCompanyProfile	169
Gambar 5.90 Kode fungsi addCompanyContact.....	169
Gambar 5.91 Kode fungsi addOrder	170
Gambar 5.92 Kode fungsi addNotification	171
Gambar 5.93 Kode fungsi addShipment	171
Gambar 5.94 Kode fungsi registerUser	172
Gambar 5.95 Kode fungsi registerDevice	173
Gambar 5.96 Kode fungsi loginGplus.....	173
Gambar 5.97 Kode fungsi loginDefault	174
Gambar 5.98 Kode fungsi forgotPassword	175
Gambar 5.99 Kode fungsi updateCatalog	176
Gambar 5.100 Kode fungsi updateDevice.....	176
Gambar 5.101 Kode fungsi updateCompanyProfile	176
Gambar 5.102 Kode fungsi updateUser	177
Gambar 5.103 Kode fungsi updateOrder.....	177
Gambar 5.104 Kode fungsi updatePassword.....	178
Gambar 5.105 Kode fungsi removeCatalogById	178

Gambar 6.2 Halaman <i>landing menu</i> untuk pengguna yang tidak <i>login</i>	181
Gambar 6.1 Splashscreen aplikasi Meme Florist	181
Gambar 6.4 Halaman kategori bunga.....	181
Gambar 6.3 Halaman detail bunga	181
Gambar 6.6 Halaman tentang kami	182
Gambar 6. 5 Halaman cara pembayaran.....	182
Gambar 6. 7 Halaman lokasi kami	182
Gambar 6. 8 Halaman hubungi kami.....	182
Gambar 6. 9 Halaman <i>Direct call phone</i>	183
Gambar 6. 10 Halaman <i>Direct copied to clipboard</i>	183
Gambar 6. 11 Halaman <i>Direct to messaging</i>	183
Gambar 6. 12 Halaman <i>Direct to email app</i>	183
Gambar 6.14 Halaman login pengguna.....	185
Gambar 6.13 Halaman <i>Direct to save to contacts</i>	185
Gambar 6.15 Halaman <i>landing menu</i> bagi pengguna yang sudah <i>login</i>	186
Gambar 6.16 Halaman akun saya.....	186
Gambar 6.18 Halaman menambah pesanan bunga ke keranjang belanja.....	188
Gambar 6.17 Halaman edit akun saya.....	188

Gambar 6.20 Halaman pemesanan bunga	190
Gambar 6.19 Halaman keranjang belanja	190
Gambar 6. 22 Halaman histori pemesanan.....	191
Gambar 6. 21 <i>Event click</i> pada histori pemesanan.....	191
Gambar 6.24 Halaman histori notifikasi	191
Gambar 6. 23 Halaman detail histori pemesanan.....	191
Gambar 6. 26 Halaman registrasi	193
Gambar 6. 25 Halaman login sebagai bukti sukses melakukan registrasi	193
Gambar 6. 28 Halaman forgot password.....	194
Gambar 6. 27 Halaman proses permintaan reset password.....	194
Gambar 6. 30 <i>Toast</i> isi pesan notifikasi	194
Gambar 6. 29 Antarmuka <i>Notification Bar device</i> untuk notifikasi <i>broadcast</i>	194
Gambar 6. 32 Antarmuka <i>Notification Bar device</i> untuk notifikasi <i>personal</i>	195
Gambar 6. 31 Halaman notifikasi pengiriman	195
Gambar 6.33 Halaman <i>Login</i> administrator	196
Gambar 6.34 Halaman beranda <i>web</i> administrator	197
Gambar 6.35 Halaman daftar katalog bunga	197
Gambar 6.36 Halaman daftar katalog bunga setelah <i>filter</i> kategori	198

Gambar 6.37 Tombol tambah di halaman daftar katalog bunga	199
Gambar 6.38 Halaman menambah data katalog bunga	199
Gambar 6.39 Tombol Sunting untuk mengubah data katalog bunga	200
Gambar 6.40 Halaman mengubah data katalog bunga	200
Gambar 6.41 Tombol hapus untuk menghapus data katalog bunga	201
Gambar 6.42 Confirm Dialog untuk menghapus katalog yang dipilih	201
Gambar 6.43 Halaman daftar pemesanan pelanggan	202
Gambar 6.44 Salah satu pemesanan pelanggan sebelum dikonfirmasi	203
Gambar 6.45 Halaman konfirmasi pemesanan pelanggan	203
Gambar 6.46 Email konfirmasi pemesanan bunga.....	204
Gambar 6.47 Salah satu pemesanan pelanggan setelah dikonfirmasi	204
Gambar 6. 49 Detail notifikasi konfirmasi pengiriman di device pelanggan	205
Gambar 6. 48 Notifikasi konfirmasi pemesanan di device pelanggan	205
Gambar 6. 51 Histori notifikasi yang masuk ke device pelanggan	205
Gambar 6. 50 Notifikasi pemberitahuan di device pelanggan...	205

Gambar 6.52 Halaman pengiriman pemberitahuan ke pelanggan	206
Gambar 6.53 Halaman pengaturan profil dan kontak Meme Florist	207
Gambar 6.54 Halaman mengubah data profil dan kontak Meme Florist	208
Gambar 6. 56 <i>Email link</i> aktivasi pelanggan baru.....	209
Gambar 6. 55 Registrasi pelanggan baru di aplikasi Meme Florist	209
Gambar 6.57 Halaman notifikasi aktivasi akun berhasil	209
Gambar 6.58 Halaman notifikasi aktivasi akun gagal	210
Gambar 6. 60 <i>Email</i> berisi <i>link</i> untuk <i>reset password</i>	211
Gambar 6. 59 Permintaan <i>forgot password</i> di aplikasi Meme Florist	211
Gambar 6.61 Halaman untuk <i>reset password</i>	211
Gambar 6.62 Halaman notifikasi <i>reset password</i> berhasil.....	212
Gambar 6.63 Halaman notifikasi permintaan <i>reset password</i> ditolak.....	212
Gambar 6.64 Halaman menampilkan daftar API di Google APIs explorer.....	213
Gambar 6.65 Halaman menampilkan daftar fungsi dari Florist API	213

DAFTAR TABEL

Tabel 4.1 Atribut <i>Kind User</i>	70
Tabel 4.2 Atribut <i>Kind Password</i>	71
Tabel 4.3 Atribut <i>Kind Category</i>	72
Tabel 4.4 Atribut <i>Kind Catalog</i>	72
Tabel 4.5 Atribut <i>Kind OrderItem</i>	73
Tabel 4.6 Atribut <i>Kind Order</i>	73
Tabel 4.7 Atribut <i>Kind Shipment</i>	74
Tabel 4.8 Atribut <i>Kind Company</i>	75
Tabel 4.9 Atribut <i>Kind CompanyContact</i>	75
Tabel 4.10 Atribut <i>Kind GCMDevices</i>	76
Tabel 4.11 Atribut <i>Kind Notification</i>	77

BAB I

PENDAHULUAN

1.1 Latar Belakang

PT. Tunas Sinergi Sejahtera atau yang dikenal dengan nama 7Langit adalah perusahaan yang bergerak dalam pembuatan aplikasi berupa teknologi perangkat bergerak (*mobile*) premium di Indonesia. 7Langit berkomitmen untuk membuat aplikasi *mobile platform* yang bisa memberikan manfaat yang besar bagi penggunanya dan perusahaan yang menggunakan jasanya [1]

Semakin berkembangnya pasar aplikasi *mobile* di Indonesia, 7Langit berkomitmen melebarkan sayap dalam menguasai pasar dunia teknologi Indonesia. Hal ini dibuktikan dengan beberapa *project* baru 7Langit yang melayani Usaha Kecil Menengah (UKM) dalam membuat aplikasi *mobile* berupa teknologi informasi usahanya. Setelah diteliti lebih jauh, aplikasi *mobile* UKM satu dengan yang lainnya ternyata memiliki beberapa kesamaan dalam fitur aplikasi didalamnya. Contohnya fitur katalog, fitur *e-cart*, fitur kontak kami, fitur tentang kami, fitur histori pemesanan, dan histori notifikasi. Untuk mempersingkat waktu penggeraan dan mengefisiensikan masa penggeraan aplikasi *mobile* UKM satu dengan yang lainnya, 7Langit mengembangkan aplikasi *mobile generic* untuk UKM pada platform Android. Harapannya, jika terdapat *client* dari UKM yang ingin membuat aplikasi *mobile* yang serupa tidak perlu mengerjakan aplikasi tersebut dari awal, namun tinggal mengimplementasikan ulang dan memperbaiki beberapa bagian untuk disesuaikan dengan keinginan UKM tersebut [2]

Pada kesempatan kerja praktik kali ini, kami diberikan tugas untuk merancang aplikasi perangkat *mobile* Android UKM dan *website* administrator *generic* dengan studi kasus berupa Toko Bunga Meme Florist yang ingin membuat aplikasi perangkat *mobile* Android. Meme Florist ingin penggunanya dapat secara mudah menemukan berbagai jenis bunga melalui aplikasi perangkat *mobile*

Android yang berisi katalog bunga, melakukan pemesanan bunga secara *online*, melakukan pemesanan bunga dengan kuantitas tertentu, mendapatkan notifikasi pengguna berupa notifikasi promosi harga bunga dan notifikasi secara *personal* mengenai pengiriman bunga yang telah dipesan. Selain aplikasi *mobile Android* kami juga diberikan tugas untuk membuat *website* untuk administrator Meme Florist secara sederhana untuk mengelola data terkait produk bunga Meme Florist.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan, didapatkan rumusan masalah sebagai berikut:

- Bagaimana cara untuk memudahkan pengguna dalam memilih bunga melalui katalog?
- Bagaimana cara untuk memudahkan pengguna dalam melakukan pemesanan bunga secara *online*?
- Bagaimana cara untuk memudahkan pengguna mendapatkan notifikasi pengiriman bunga secara *personal* dan notifikasi promosi harga bunga?
- Bagaimana cara untuk memudahkan pengguna melakukan pengaturan untuk penerimaan notifikasi pada *device*?
- Bagaimana cara untuk memudahkan administrator dalam mengelola data terkait produk melalui *website*?
- Bagaimana cara untuk memudahkan administrator dalam memantau data terkait pemesanan bunga oleh pengguna?
- Bagaimana cara untuk memudahkan administrator dalam memberikan notifikasi pengiriman bunga secara *personal* dan notifikasi promosi harga bunga pada pengguna?
- Bagaimana cara untuk memudahkan administrator dalam mengatur informasi tentang profil dan kontak Meme Florist?

- Bagaimana cara untuk memudahkan administrator dalam memberikan konfirmasi aktivasi akun, konfirmasi permintaan *password*, dan konfirmasi pemesanan?

1.3 Tujuan dan Manfaat

Berdasarkan latar rumusan masalah, didapatkan rumusan masalah sebagai berikut:

- Untuk memudahkan pengguna dalam memilih bunga melalui katalog.
- Untuk memudahkan pengguna dalam melakukan pemesanan bunga secara *online* dengan kuantitas tertentu.
- Untuk memudahkan pengguna mendapatkan notifikasi pengiriman bunga secara *personal* dan notifikasi promosi harga bunga.
- Untuk memudahkan pengguna melakukan pengaturan dalam penerimaan notifikasi.
- Untuk memudahkan administrator dalam mengelola data terkait produk melalui *website*.
- Untuk memudahkan administrator dalam memantau data terkait pemesanan bunga oleh pengguna.
- Untuk memudahkan administrator dalam memberikan notifikasi pengiriman bunga secara *personal* dan notifikasi promosi harga bunga pada pengguna.
- Untuk mempermudah administrator dalam mengatur informasi tentang profil dan kontak Meme Florist.
- Untuk memudahkan administrator dalam memberikan konfirmasi aktivasi akun, konfirmasi permintaan *password*, dan konfirmasi pemesanan.

1.4 Waktu Pelaksanaan

kerja praktik dilaksanakan selama satu bulan, terhitung sejak tanggal 23 Juni 2014 sampai dengan tanggal 23 Juli 2014.

1.5 Sistematika Penulisan

Laporan Kerja Praktik ini dibagi menjadi 7 bab dengan rincian sebagai berikut:

- **Bab I : Pendahuluan**
Pada bab ini dijelaskan latar belakang, permasalahan yang dihadapi, tujuan, batasan masalah yang dihadapi, waktu pelaksanaan kerja praktik, serta sistematika penulisan laporan kerja praktik.
- **Bab II : Sejarah dan Profil Perusahaan**
Bab ini berisi sekilas mengenai PT. Tunas Sinergi Sejahtera.
- **Bab III : Tinjauan Pustaka**
Pada bab ini dijelaskan tentang tinjauan pustaka yang digunakan untuk menyelesaikan aplikasi yang dibuat.
- **Bab IV : Analisa Perancangan**
Pada bab ini dijelaskan mengenai perancangan antarmuka aplikasi serta perancangan data yang digunakan dalam aplikasi. Selanjutnya dijelaskan pula implementasinya.
- **Bab V: Implementasi**
Bab ini menjelaskan uraian mengenai implementasi pada pengembangan aplikasi *mobile generic* dan *website generic* dengan menggunakan studi kasus Toko Bunga Meme Florist.
- **Bab VI: Uji Coba dan Evaluasi**
Bab ini berisi hasil uji coba dan evaluasi dari perangkat lunak yang dikembangkan selama pelaksanaan kerja praktik.
- **Bab VII: Kesimpulan dan Saran**
Bab ini berisi kesimpulan dan saran dari proses pelaksanaan tugas praktik

BAB II

PROFIL PERUSAHAAN

2.1 Sejarah Perusahaan

PT. Tunas Sinergi Sejahtera atau yang dikenal dengan 7Langit adalah perusahaan *software house* industri digital kreatif yang bergerak di aplikasi *mobile* premium yang berlokasi di Jakarta. 7Langit berkomitmen untuk membuat aplikasi *mobile platform* yang bisa memberikan manfaat yang besar bagi penggunanya dan perusahaan yang menggunakan jasanya. Sebagai salah satu perusahaan aplikasi *mobile* di Indonesia, dengan bangga 7Langit mendukung kampanye “Menjadi tuan rumah di negeri sendiri” dalam pembuatan aplikasi *mobile* dengan kualitas standar internasional [1]. Beberapa perusahaan yang telah bekerja sama dengan 7Langit diantaranya adalah PT. XL Axiata Tbk, PT. Telkomsel, PT. Astra Honda Motor, PT. Nestle Indonesia, Java Festival Production, Blood for Life Indonesia dan Jakarta Marketing Week. Berdasarkan perusahaan yang telah bekerja sama dengan 7Langit tersebut, aplikasi *mobile* yang dikembangkan antara lain aplikasi bisnis yang berbasis teknologi informasi, aplikasi *social project*, aplikasi *augmented reality* dan aplikasi *mobile game* [3].

Pada awalnya, 7Langit dirintis pada tahun 2009 oleh Titi Rusdi dan Oon Arfiandwi yang saat itu berteman baik meskipun keduanya memiliki latar belakang pendidikan yang berbeda. Oon Arfiandwi yang memiliki latar belakang pendidikan Teknik Informatika membuat aplikasi *mobile* Teman Ibadah berbasis platform Blackberry, kemudian Titi Rusdi yang memiliki latar belakang pendidikan Ekonomi menjualkan produk aplikasi yang telah dibuat tersebut dan menawarkan jasa pembuatan aplikasi *mobile* ke beberapa perusahaan. Karena Teman Ibadah tidak diminati oleh perusahaan-perusahaan tersebut, akhirnya Oon Arfiandwi dan Titi Rusdi memutuskan untuk melepaskan Teman Ibadah untuk bisa diunduh secara gratis oleh pengguna Blackberry. Saat itu kebetulan

adalah bulan Ramadhan, sehingga tidak disangka jumlah pengunduh aplikasi Teman Ibadah naik secara drastis hingga menembus 30.000 pengunduh. Begitu melihat tingginya jumlah pengunduh, keduanya sepakat melepas aplikasi Teman Ibadah sebagai *project social* pertama bagi 7Langit dan menjadikannya sebagai portofolio untuk jasa pembuatan aplikasi *mobile* ke perusahaan-perusahaan. Sejak saat itu, *project* baru pun lantas berdatangan ke 7Langit dan dimulailah proses rekrutmen tim pengembang untuk pengerjaan seluruh *project* tersebut [4] .

Dengan SDM yang seluruhnya masih berusia muda, 7Langit tetap mampu menghasilkan karya – karya yang semakin baik melalui proses inovasi dan *improvement* tanpa henti.

2.2 Logo Perusahaan

Berikut adalah logo dari perusahaan PT. Tunas Sinergi Sejahtera (7Langit) seperti yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 Logo 7Langit

2.3 Visi dan Misi Perusahaan

Visi:

Menuju Indonesia yang lebih baik melalui teknologi.

Misi:

Menjadi tuan rumah di negeri sendiri melalui aplikasi *mobile* premium berstandar internasional.

BAB III

TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan mengenai dasar teori yang digunakan selama proses penggerjaan dan pengembangan aplikasi perangkat *mobile* Android untuk pemesanan bunga *online*. Dasar teori yang akan dibahas adalah mengenai Android, keunggulan Android, Android SDK, bahasa pemrograman Java dan XML, *Google Coud Messaging* (GCM), Google Maps, Google App Engine, Google Cloud Endpoints, Java Datastore API, *Java Data Objects* (JDO) with AppEngine, Blobstore Java API, dan Apache Velocity.

3.1 Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk telepon seluler (*smartphone*) maupun tablet. Android pada awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, kemudian di tahun 2005 Google membelinya secara resmi untuk dikembangkan lebih lanjut. Bahkan kini, Android sudah mulai dirancang menjadi OS bagi *smartwatch* dan Google Glass oleh Google. Kelebihan dari sistem operasi Android ialah tersedianya *platform* terbuka (*open source*) bagi para pengembang untuk menciptakan aplikasi-aplikasi baru yang nantinya dapat digunakan oleh berbagai macam piranti bergerak.

Perkembangan sistem operasi Android tergolong pesat, hal ini dibuktikan dengan adanya berbagai macam versi yang dimiliki oleh sistem operasi ini [5] Adapun versi Android yang telah dirilis antara lain:

- a) Android versi 1.0 (Astro)
- b) Android versi 1.1 (Bender)
- c) Android versi 1.5 (Cupcake)
- d) Android versi 1.6 (Donut)
- e) Android versi 2.0/2.1 (Eclair)
- f) Android versi 2.2 Frozen Yoghurt (Froyo)
- g) Android versi 2.3 (Gingerbread)
- h) Android versi 3.0/3.1 (HoneyComb)
- i) Android versi 4.0 (Ice Cream Sandwich)

- j) Android versi 4.1.x,4.2.x,4.3.x (Jelly Bean)
- k) Android versi 4.4 (KitKat)
- l) Android versi 5.0 (L)

3.2 Keunggulan Android

Keunggulan yang dimiliki Android [6] dibandingkan sistem operasi yang lain adalah sebagai berikut:

- **Pasar yang Luas**
Pengembang Android memiliki kesempatan yang luas untuk mengembangkan aplikasi berbasis Android karena pengguna perangkat Android selalu bertambah dan pasarnya meledak setiap harinya.
- **Open Platform**
Sistem operasi Android adalah *open platform*. Hal ini berarti tidak mengikat satu perangkat keras dan atau satu *provider* dalam pengembangan Android. Sehingga semua pengembang yang ingin mengembangkan aplikasi Android dapat secara mudah mendapatkan *source code* untuk pengembangannya lebih lanjut. Hal inilah yang menjadikan Android memiliki pasar yang luas dan memiliki pengguna yang secara signifikan naik dibandingkan sistem operasi perangkat bergerak yang lainnya.
- **Cross-compatibility**
Android dapat berjalan di banyak perangkat dengan ukuran layar dan resolusi yang berbeda-beda. Android pada dasarnya telah menyediakan *tools* yang dapat membantu penggunanya mengembangkan aplikasi yang berbasis *cross-compatibility*. Google mengijinkan aplikasi yang ada untuk berjalan hanya pada perangkat yang *compatible*. Misalnya jika suatu aplikasi membutuhkan kamera depan, maka perangkat yang memiliki kamera depan saja yang bisa melihat aplikasi tersebut di *Android Market*.

- **Dapat mengkombinasikan lebih dari satu service**

Android memberikan pengembang berupa fasilitas untuk mengkombinasikan lebih dari satu *service* yang dimiliki Android untuk memaksimalkan aplikasi yang dibuat oleh pengembang. Hal ini dapat dilakukan dengan semua API yang telah disediakan oleh Android, sehingga sangat mudah untuk menggunakan dua atau lebih fitur yang terdapat di Android dikolaborasikan ke dalam satu aplikasi. Contoh fitur Android yang pernah dikolaborasikan adalah fitur *geolocation* dan *social networking*, *geolocation* dan *gaming*, *contacts* dan *internet*.

3.3 Android SDK

Android SDK mencakup perangkat *tools* pengembangan yang komprehensif. Android SDK terdiri dari *debugger*, *libraries*, *handset emulator*, dokumentasi, contoh kode program dan tutorial. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk *desktop modern*), Mac OS X 10.5.8 atau lebih, Windows XP (32-bit), Windows Vista (64-bit), dan Windows 7 (64-bit) [6] Persyaratan mencakup JDK, Apache Ant dan Python 2.2 atau lebih. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan plugin *Android Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit dokumen Java dan XML (*Extensible Markup Language*) serta menggunakan peralatan Command Line untuk menciptakan, membangun, melakukan *debug* aplikasi Android dan pengendalian perangkat Android (misalnya *reboot*, memasang paket perangkat lunak).

3.4 Bahasa Pemrograman Java dan XML

Aplikasi yang berbasis Android dikembangkan dengan bahasa pemrograman Java. Saat ini, Java adalah satu-satunya pilihan jika pengembang ingin membuat aplikasi Android secara *native*. Java adalah bahasa pemrograman yang sangat populer dikembangkan oleh Sun Microsystems (yang sekarang dimiliki oleh Oracle). Pengembangan yang sangat panjang setelah C dan C++, Java termasuk dalam bahasa pemrograman yang *powerful* seperti halnya *library* yang dimilikinya. Inti pentingnya Java adalah mudah dipahami dan mudah dipelajari, didesain menjadi *platform-independent* dan aman menggunakan *virtual machines*, dan berorientasi objek [7]

Android juga menggunakan bahasa XML untuk mengatur tampilan aplikasi, seperti mengatur letak posisi tombol, tombol background, posisi teks, dan hampir semua yang berhubungan dengan *layout*.

3.5 Google Cloud Messaging (GCM)

Google Cloud Messaging (GCM) untuk Android adalah sebuah *service* yang mengijinkan kita mengirimkan data dari *server* ke pengguna Android yang *compatible*, dan menerima pesan dari perangkat yang memiliki koneksi yang sama dengan GCM *service* dengan melakukan antrian pesan dan dikirimkan ke target aplikasi Android yang sedang berjalan di perangkat target. GCM seutuhnya gratis sebesar apapun ukuran pesan yang akan dikirimkan maupun diterima dan tidak ada batasan kuota dalam penggunaannya [8]

Untuk menggunakan GCM, setiap perangkat yang akan dilibatkan harus terdaftar dalam *webserver* GCM. Hal ini berguna untuk menyimpan registrasi GCM ID masing-masing perangkat ke *webserver* dan satu perangkat hanya terdaftar satu registrasi GCM

ID saja. Setiap aplikasi Android yang menggunakan GCM, di dalam *source code* aplikasi tersebut telah dimasukkan fungsi GCM yang berisi inisialisasi registrasi GCM ID jika perangkat tersebut belum pernah didaftarkan sebelumnya, dan jika sudah terdaftar maka akan menyimpan nomor registrasi GCM ID perangkat tersebut menjadi *session* selama aplikasi tersebut berjalan. Tujuannya adalah untuk memanipulasi notifikasi pada tata kelola profil pengguna. Jika pengguna ingin mendapatkan notifikasi, maka dapat mengubah inisialisasi GCM notifikasi pengguna menjadi *true*, sedangkan jika pengguna tidak ingin mendapatkan notifikasi, maka dapat mengubah inisialisasi GCM notifikasi pengguna menjadi *false* [9]

Device yang didaftarkan hanya dapat menerima notifikasi saja dari server. Teknis penggunaannya adalah sebagai berikut:

- Perangkat mengirimkan permintaan ke GCM untuk melakukan pemeriksaan apakah perangkat tersebut sudah terdaftar dalam *database* GCM atau belum.
- Kemudian jika belum maka GCM akan memberikan konfirmasi kepada perangkat untuk mendaftarkan perangkat ke *webserver* untuk memberikan registrasi GCM ID dan menyimpannya di *database* GCM.
- Registrasi GCM ID tersebut kemudian dikembalikan pada perangkat untuk menginisialisasi *session* GCM pada aplikasi.
- Untuk mengirimkan notifikasi kepada pengguna melalui GCM, *webserver* melakukan pemeriksaan di *database* untuk mendapatkan registrasi GCM ID kemudian mengirimkan permintaan pengiriman notifikasi ke GCM beserta registrasi GCM ID dan pesan.
- *Device* yang memiliki registrasi GCM ID yang telah dipilih tersebut akan mendapatkan notifikasi pesan dari *server* yang akan muncul pada *notification bar device*.

3.6 Google Maps

Sebagai perangkat bergerak yang memiliki tingkat mobilitas yang cukup tinggi, Android memberikan fitur menarik mengenai *geolocation* yang dapat dimanfaatkan pada perangkat bergerak Android sebagai pendekripsi lokasi pengguna sekarang berada maupun lokasi yang ingin dijangkau oleh pengguna. Android menyediakan API yang secara terbuka mudah diakses siapapun untuk digunakan pengembang dalam melakukan kolaborasi fitur di dalam aplikasinya. Fitur di dalam Google Maps yang dapat digunakan pada Android antara lain memberikan *marker* lokasi berdasarkan petunjuk *latitude* dan *longitude*, pemberian nama pada *marker*, dan melakukan *zoom in* pada daerah *marker* tersebut sehingga dapat menjadi titik fokus pengguna dalam mengetahui titik lokasi *marker*. (<https://developers.google.com/>)

3.7 Google App Engine

Google App Engine adalah *Platform as a Service* (PaaS) buatan Google yang menyediakan sekumpulan API (*Application Programming Interface*), yang memungkinkan pengembang untuk membuat dan menjalankan aplikasi – aplikasi berbasis *web* dengan fasilitas *hosting* di *server* Google itu sendiri. Google App Engine ini tersedia secara gratis, dengan kapasitas penyimpanan untuk semua aplikasi sampai dengan 1 GB dengan CPU yang cukup dan *bandwidth* untuk mendukung sebuah aplikasi yang efisien untuk melayani tampilan sekitar 5 juta halaman per bulan. Google App Engine ini mendukung aplikasi yang dibuat dalam beberapa bahasa pemrograman, seperti Java, Python, PHP, dan GO [10] . Beberapa keunggulan dari Google App Engine ini antara lain:

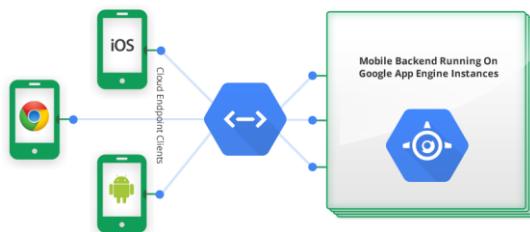
- *Persistent storage* yang dapat diakses dengan *query*, *sorting*, dan *transaction*.
- Dapat secara otomatis skalabilitasnya dan *load balancing*.

- Memiliki *asynchronous task queues* yang memungkinkan untuk bekerja di luar lingkup *web request*.
- *Scheduled tasks* untuk memicu beberapa *event* dalam waktu yang ditentukan ataupun jangka waktu tertentu.
- Memiliki integrasi dengan *Google Cloud Service and APIs* yang lain.

3.8 Google Cloud Endpoints

Google Cloud Endpoints terdiri dari beberapa *tools*, *library* dan kemampuan yang dapat digunakan untuk membangkitkan API (*Application Programming Interface*) dan *client library* serta sebuah aplikasi App Engine, merujuk pada sebuah *API backend*, untuk menyederhanakan pengaksesan data oleh klien dari aplikasi yang lain seperti yang ditunjukkan pada Gambar 3.1. Endpoints dapat memudahkan untuk pengembang membuat sebuah aplikasi *web backend* untuk *web client* dan *mobile client* seperti Android ataupun iOS milik Apple [11].

Untuk pengembang *mobile*, Endpoints menyediakan cara yang sederhana untuk mengembangkan sebuah *shared web backend* dan juga infrastruktur penting seperti autentikasi OAuth 2.0.



Gambar 3.1 Arsitektur Google Cloud Endpoints

3.9 Java Datastore API

Datastore adalah sebuah layanan penyimpanan data yang bersifat *schemaless NoSQL* dan *scalable storage* untuk aplikasi *web*. Java Datastore SDK menyertakan pula implementasi dengan *Java Data Object* (JDO) dan *Java Persistence API* (JPA). Datastore menyimpan objek data yang kemudian dikenal sebagai *entity* [12]. Sebuah *entity* dapat memiliki satu atau lebih properti, dimana terdapat beberapa atribut dengan beberapa tipe data berbeda-beda yang mendukung (dapat berupa *string*, *integer*, ataupun yang lain). Setiap *entity* dikenali dari *kind* yang mengategorikan masing-masing *entity* sesuai dengan tujuan *query* dan *key* yang secara unik sebagai identitas dalam *kind*.

Datastore memiliki beberapa keunggulan yang antara lain:

- Tidak mudah *down*
- Transaksi dapat dilakukan secara spesifik (*atomic*)
- Ketersediaan tinggi untuk kebutuhan pembacaan dan penulisan
- Konsistensi yang kuat untuk pembacaan dan *ancestor queries*

3.10 Java Data Objects (JDO) with AppEngine

Java Data Objects (JDO) adalah sebuah antar muka standar untuk penyimpanan objek yang mengandung data ke dalam basis data. Standar disini mendefinisikan antar muka untuk menambahkan objek data, menerima objek dengan *query*, dan berinteraksi dengan basis data melalui transaksi [13]. Sebuah aplikasi yang menggunakan antar muka JDO dapat berkerja di beberapa jenis basis data yang berbeda tanpa menggunakan kode spesifik dari masing-masing basis data tersebut, termasuk diantaranya adalah basis data berdasarkan relasi/hubungan, basis data berdasarkan hirarki, dan basis data berdasarkan objek. Seperti

antar muka standar yang lain, JDO memudahkan untuk menjembatani aplikasi di solusi penyimpanan (basis data) yang berbeda-beda.

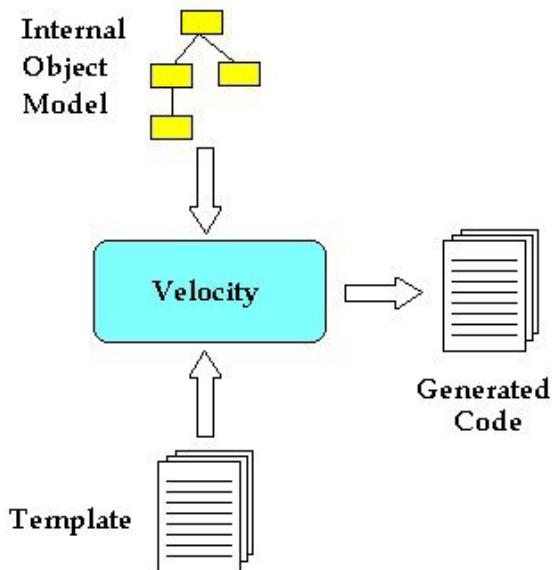
3.11 Blobstore Java API

Blobstore API memungkinkan sebuah aplikasi untuk menyediakan objek data, kemudian disebut dengan *blob*, yang mana memiliki ukuran lebih besar daripada objek biasanya yang diperbolehkan disimpan di dalam layanan Datastore [14]. *Blob* sangat berguna untuk menyediakan berkas dengan ukuran besar seperti gambar, dan juga untuk memungkinkan pengguna mengunggah berkas lainnya dengan ukuran yang besar. *Blob* dibuat dengan proses pengunggahan berkas melalui *HTTP request*. Biasanya, aplikasi akan melakukan hal ini dengan mengirimkan sebuah *form* dengan *file upload field* di dalamnya. Ketika pengguna mengirimkannya, maka Blobstore akan membuat sebuah *blob* dari isi berkas tersebut dan mengembalikannya dalam bentuk referensi kabur kepada *blob*, yang disebut dengan *blob key* dan nantinya digunakan untuk menyediakan *blob* yang diminta.

3.12 Apache Velocity

Apache Velocity adalah sebuah *template engine* berbasis Java yang dapat melakukan *data rendering* dari objek Java biasa menjadi teks, XML, email, SQL, Post Script, HTML, dan lain sebagainya [15]. Sintaks dari *template* dan *rendering engine*, keduanya mudah untuk dipahami dan cepat untuk dipelajari dan diimplementasikan. Dengan Velocity ini, *programmer* akan bisa fokus mengerjakan kode sumber yang fungsional, sementara itu secara paralel *template designer* pula secara langsung dapat memodifikasi *template* untuk membuat tampilan yang semenarik mungkin. Dalam sebuah aplikasi *web*, pemisahan *Model-View-*

Controller (MVC) dapat dilakukan karena *template* tidak mengandung kode sumber fungsional seperti yang ditunjukkan pada Gambar 3.2, namun hanya berfungsi sebagai tampilan dari hasil olahan kode fungsional.



Gambar 3.2 Skema Cara Kerja Apache Velocity

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai analisis kebutuhan dari aplikasi yang dibangun serta detil dari perancangan fungsional sistem, data, maupun antarmuka.

4.1. Analisis dan Perancangan Aplikasi *Mobile*

Pada subab ini akan dijelaskan mengenai analisis dan perancangan aplikasi Meme Florist pada perangkat *mobile*.

4.1.1. Analisis Kebutuhan

Secara umum spesifikasi kebutuhan perangkat lunak yang akan dikembangkan dalam aplikasi perangkat *mobile* Android yang seterusnya akan diberi nama aplikasi Meme Florist ini adalah sebagai berikut:

- a. **Menampilkan kategori bunga dan masing-masing anggota kategori bunga melalui katalog**
Dapat menampilkan kategori bunga dan detail masing-masing anggota kategori bunga tersebut yang telah disimpan pada Datastore berdasarkan ID kategori yang dipilih.

- b. **Menampilkan detail bunga pada kategori tertentu berdasarkan tingkat harga yang dipilih**
Dapat menampilkan urutan detail bunga pada kategori tertentu berdasarkan harga bunga dari tingkat paling murah dan tingkat paling mahal yang telah disimpan di Datastore. Data detail masing-masing bunga yang ditampilkan adalah gambar bunga, kode bunga, ukuran bunga dalam sentimeter, harga bunga, dan deskripsi.

- c. Menyimpan data keranjang belanja pengguna**
Dapat menyimpan detail bunga yang dipilih pengguna ke dalam keranjang belanja untuk disimpan sementara di *device*. Data keranjang belanja yang disimpan adalah *list* objek bunga, yang masing-masing memiliki data detail bunga yaitu gambar bunga, kode bunga, ukuran bunga dalam satuan sentimeter, harga bunga, deskripsi, kuantitas bunga, dan pesan pengirim untuk penerima. Penyimpanan data pemesanan diikuti dengan penyimpanan data detail bunga.

- d. Menyimpan data pemesanan pengguna**
Dapat menyimpan data keranjang belanja pengguna pada Datastore. Data pemesanan yang disimpan adalah *list* objek bunga, yang masing-masing memiliki data detail bunga yaitu gambar bunga, kode bunga, ukuran bunga dalam satuan sentimeter, harga bunga, deskripsi, kuantitas bunga, dan pesan pengirim untuk penerima.

- e. Menampilkan seluruh data keranjang belanja pengguna**
Dapat menampilkan data keranjang belanja pengguna yang telah tersimpan sementara di *device*. Data detail keranjang belanja yang ditampilkan adalah *list* objek bunga yang telah dipilih oleh pengguna. Masing-masing memiliki data detail bunga yaitu gambar bunga, kode bunga, ukuran bunga dalam satuan sentimeter, harga, dan kuantitas bunga yang dibeli.

- f. Menghapus bunga pada data pemesanan pengguna**
Dapat menghapus bunga dan detail bunga pada data pemesanan pengguna yang disimpan sementara di *device* sesuai dengan objek bunga yang dipilih.
- g. Menampilkan histori pemesanan dan detail histori pemesanan yang dimiliki pengguna**
Dapat menampilkan histori pemesanan dan detail histori pemesanan yang dimiliki pengguna yang telah tersimpan di Datastore. Histori pemesanan ini berisi nama tujuan pengiriman, alamat tujuan pengiriman, isi pesan pengiriman bunga, total harga pemesanan, dan status pengiriman. Detail histori pemesanan berisi mengenai detail bunga yang dibeli, terdiri dari gambar bunga, kategori bunga, kode bunga, ukuran bunga dalam satuan sentimeter, harga bunga, dan kuantitas pemesanan bunga.
- h. Menampilkan histori notifikasi dari Meme Florist**
Dapat menampilkan histori notifikasi dari Datastore berdasarkan objek pengguna yang melakukan *login*. Histori notifikasi merupakan kumpulan seluruh notifikasi yang pernah didapatkan oleh pengguna yang telah terdaftar melalui registrasi, baik berupa notifikasi promosi maupun notifikasi secara *personal* mengenai status pengiriman bunga.

-
- i. **Menampilkan data diri pengguna**
Dapat menampilkan data diri pengguna yang disimpan di Datastore berdasarkan objek pengguna yang melakukan *login*.
 - j. **Memperbarui data diri pengguna**
Dapat memperbarui data diri pengguna yang lama dengan yang baru lalu menyimpannya kembali di Datastore, kecuali data diri pengguna berupa *email*. Karena *email* menjadi *field* unik untuk setiap pengguna sehingga tidak bisa diperbarui.
 - k. **Menampilkan informasi tentang Meme Florist**
Dapat menampilkan informasi mengenai Meme Florist yang telah disimpan di Datastore. Informasi Meme Florist terdiri dari deskripsi Meme Florist, cara pembayaran setelah melakukan pemesanan bunga dan lokasi Meme Florist yang ditunjukkan melalui Google Maps.
 - l. **Menampilkan informasi mengenai kontak Meme Florist**
Dapat menampilkan informasi mengenai kontak Meme Florist yang telah disimpan di Datastore. Kontak Meme Florist ini terdiri dari seluruh jenis nomor yang bisa dihubungi seperti BBM, WhatsApp, telepon, SMS, dan *email*. Untuk kontak BBM, WhatsApp, dan *email* dapat di *copy* dan *paste* di *clipboard*. Untuk telepon dapat langsung dihubungkan dengan *call phone* dan untuk SMS dapat langsung dihubungkan dengan *send message*. Seluruh kontak tersebut juga diberikan pilihan untuk disimpan ke dalam *device* ataupun kartu SIM.

m. Menyimpan data pengguna yang telah melakukan registrasi

Dapat menyimpan data pengguna yang telah melakukan registrasi ke dalam Datastore. Data registrasi berisi mengenai nama lengkap, *email*, *password*, nomor telepon, dan alamat.

n. Menampilkan notifikasi personal

Dapat menampilkan notifikasi yang dikirimkan oleh administrator Meme Florist pada *notification bar* pada *device* pengguna. Ketika notifikasi tersebut dilakukan *event* klik, maka secara otomatis akan langsung membuka aplikasi dan menampilkan notifikasi pengiriman bunga yang telah dilakukan Meme Florist terhadap pesanan pengguna. Kemudian status pengiriman pesanan pengguna di histori pemesanan yang awalnya “bunga belum dikirim” akan berubah menjadi “bunga telah dikirim”.

o. Menampilkan notifikasi promosi produk

Dapat menampilkan notifikasi yang dikirimkan oleh administrator Meme Florist pada *notification bar* pada *device* pengguna. Ketika notifikasi tersebut dilakukan *event* klik, maka secara otomatis akan langsung membuka aplikasi dan menyimpan notifikasi tersebut ke dalam histori notifikasi.

4.1.2. Definisi Umum Aplikasi

Aplikasi Meme Florist ini dibuat untuk konsumen Meme Florist yang ingin melakukan pemesanan bunga

secara *online*. Tujuan dari aplikasi ini adalah untuk memudahkan konsumen Meme Florist dalam memilih bunga dan memesan bunga melalui katalog yang disediakan dimanapun berada. Untuk mengakses aplikasi ini, harus dipastikan *device* yang digunakan konsumen harus selalu terkoneksi dengan *internet*, karena seluruh data disimpan di Datastore Google App Engine. Sehingga untuk mengaksesnya akan membutuhkan koneksi *internet*.

Pengguna yang belum melakukan *login* dapat menggunakan fitur katalog untuk melihat kategori-kategori bunga yang dimiliki Meme Florist, kemudian pengguna dapat melihat *list* bunga dari masing-masing kategori dengan detil bunga beserta gambar bunga. Pada katalog *list* bunga dapat pula dilakukan pengurutan harga bunga dari yang termurah ataupun termahal. Selain itu, pengguna yang belum melakukan *login* tersebut dapat melihat informasi mengenai Meme Florist yang berisi deskripsi Meme Florist, lokasi Meme Florist yang ditunjukkan melalui Google Maps dan cara pembayaran melalui transfer ATM setelah melakukan pemesanan. Terakhir, pengguna tersebut dapat melihat kontak Meme Florist yang berisi semua jenis kontak yang dimiliki Meme Florist seperti BBM, WhatsApp, telepon, SMS, dan *email*.

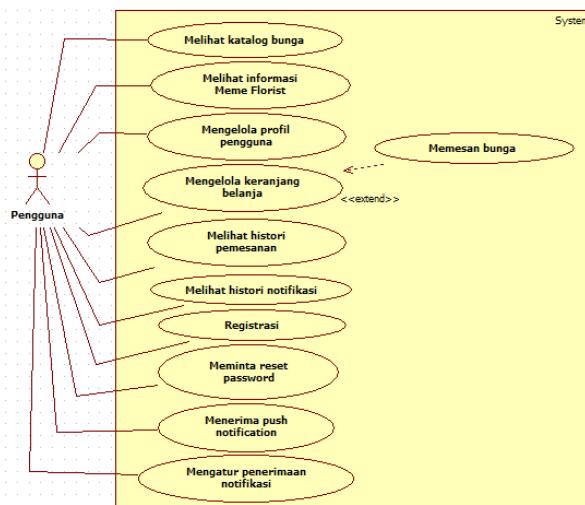
Sebelum melakukan pemesanan bunga, konsumen harus telah terdaftar menjadi pengguna aplikasi Meme Florist yang dilakukan melalui registrasi. Setelah itu, pengguna dapat melakukan *login* yang kemudian dapat mengakses lebih banyak fitur didalamnya seperti katalog, tentang Meme Florist, kontak Meme Florist, profil pengguna, histori pemesanan, histori notifikasi, dan keranjang belanja. Seluruh data yang ditampilkan disimpan di Datastore Google App

Engine, kecuali keranjang belanja. Karena terdapat kemungkinan pengguna membatalkan bunga yang dipesan. Hal ini dapat menyebabkan Datastore akan selalu berubah-ubah dan agar tidak selalu melakukan koneksi dengan *server* Google App Engine, maka dilakukan penyimpanan data keranjang belanja di *device*. Setelah pengguna memastikan pesanannya sudah benar, maka pengguna mengisi beberapa data untuk proses pengiriman bunga dan data pemesanan disimpan di Google App Engine secara permanen.

4.1.3. Perancangan Fungsionalitas Aplikasi

- **Perancangan Fungsionalitas Aplikasi**

Berdasarkan uraian di atas, gambaran spesifikasi kebutuhan dapat digambarkan dengan diagram kasus penggunaan berikut pada Gambar 4.1.



Gambar 4.1 Kasus Penggunaan Aplikasi Mobile Meme Florist

- **Cara Kerja Aplikasi**

Berikut adalah cara kerja aplikasi Meme Florist yang seluruh fiturnya berkaitan langsung dengan pengguna:

1. **Melihat katalog bunga**

Kasus penggunaan ini digunakan untuk pengguna melihat seluruh kategori bunga dan melihat detail bunga. Detail bunga berisi gambar bunga, kode bunga, ukuran bunga dalam satuan sentimeter, dan harga bunga.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin a, b]

2. **Registrasi**

Kasus penggunaan ini digunakan untuk pengguna melakukan registrasi agar mendapatkan *username* dan *password* untuk *login* ke dalam aplikasi Meme Florist. Karena fitur pemesanan bunga Meme Florist hanya disediakan untuk pengguna yang telah memiliki hak akses sebagai anggota.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin m]

3. **Meminta reset password**

Kasus penggunaan ini digunakan untuk pengguna melakukan permintaan *reset password* dengan mengisi *field email* pengguna untuk dapat mengganti *password* yang lama dengan yang baru dari *link* yang disediakan ketika mendapatkan *email* dari *server Google App Engine*.

[Spesifikasi Pengembangan yang terkait: tidak ada]

4. Melihat informasi Meme Florist

Kasus penggunaan ini digunakan untuk pengguna melihat informasi tentang Meme Florist yang berisi lokasi Meme Florist dengan Google Maps, cara membayar jika telah melakukan pemesanan bunga, dan kontak Meme Florist yang dapat dihubungi.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin k, l]

5. Login Pengguna Meme Florist

Pengguna melakukan *login* ke dalam aplikasi Meme Florist dan mendapatkan hak aksesnya sebagai pelanggan.

[Spesifikasi Pengembangan yang terkait: tidak ada]

6. Memesan bunga

Kasus penggunaan ini digunakan untuk pengguna melakukan pemesanan bunga dengan mengisi data diri penerima bunga yang telah dipesan. Sebelum melakukan pemesanan, pengguna harus telah memiliki pesanan bunga di dalam keranjang belanja.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin d]

7. Mengelola keranjang belanja

Kasus penggunaan ini digunakan untuk pengguna mengelola keranjang belanja dengan menambah bunga, memperbarui kuantitas pemesanan bunga, dan menghapus bunga yang telah dipilih sebelumnya. Pengguna menambah bunga di keranjang belanja dengan cara memilih bunga kemudian mengisi kuantitas bunga dan isi pesan untuk bunga tersebut. Pengguna memperbarui kuantitas bunga di keranjang belanja dengan cara menambah atau mengurangi kuantitas bunga yang telah dipilih sebelumnya.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin c, e, f]

8. Mengelola profil pengguna

Kasus penggunaan ini digunakan untuk pengguna dalam mengelola profil pengguna yaitu dengan melihat profil dan memperbarui profil. Untuk memperbarui seluruh data profil pengguna, hanya email pengguna tidak bisa diperbarui karena menjadi ID unik untuk masing-masing pengguna yang telah terdaftar menjadi anggota.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin i, j]

9. Melihat histori pemesanan dan detil histori pemesanan

Kasus penggunaan ini digunakan untuk pengguna melihat histori pemesanan dan detail histori pemesanan yang dimiliki pengguna. Histori

pemesanan ini berisi nama tujuan pengiriman, alamat tujuan pengiriman, isi pesan pengiriman bunga, total harga, dan status pesanan. Detail histori pemesanan adalah detail bunga yang dipesan dan berisi gambar bunga, kategori bunga, kode bunga, ukuran bunga dalam satuan sentimeter, harga per satuan bunga, dan kuantitas pembelian bunga.

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin g, n]

10. Melihat histori notifikasi

Kasus penggunaan ini digunakan untuk menampilkan histori notifikasi yang dimiliki pengguna dan berasal dari administrator Meme Florist yang berisi mengenai promosi harga bunga dan informasi terkait Meme florist. Notifikasi yang digunakan adalah *push notification* menggunakan *Google Cloud Messaging* (GCM).
[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin g, h]

11. Menerima *push notification* dari Meme Florist

Kasus penggunaan ini digunakan untuk menerima notifikasi baik itu *broadcast* maupun *personal* dari Meme Florist. Notifikasi yang digunakan adalah *push notification* menggunakan *Google Cloud Messaging* (GCM).

[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin n, o]

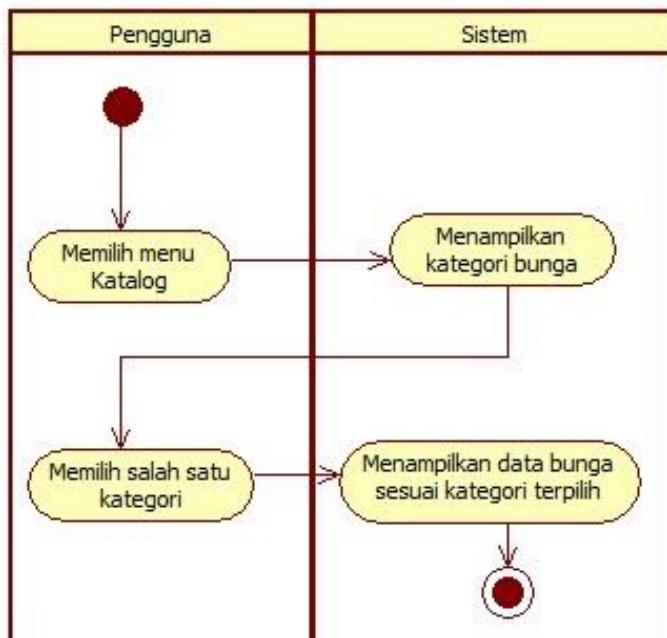
12. Melakukan pengaturan penerimaan *push notification*

Kasus penggunaan ini digunakan untuk melakukan pengaturan penerimaan *push notification* dari Meme Florist mengenai informasi promosi produk bunga. Pengguna dalam hal ini dapat melakukan pengaturan untuk mengaktifkan penerimaan notifikasi atau melakukan nonaktif notifikasi pada *device* yang dimilikinya. Notifikasi yang digunakan adalah *push notification* menggunakan *Google Cloud Messaging* (GCM).

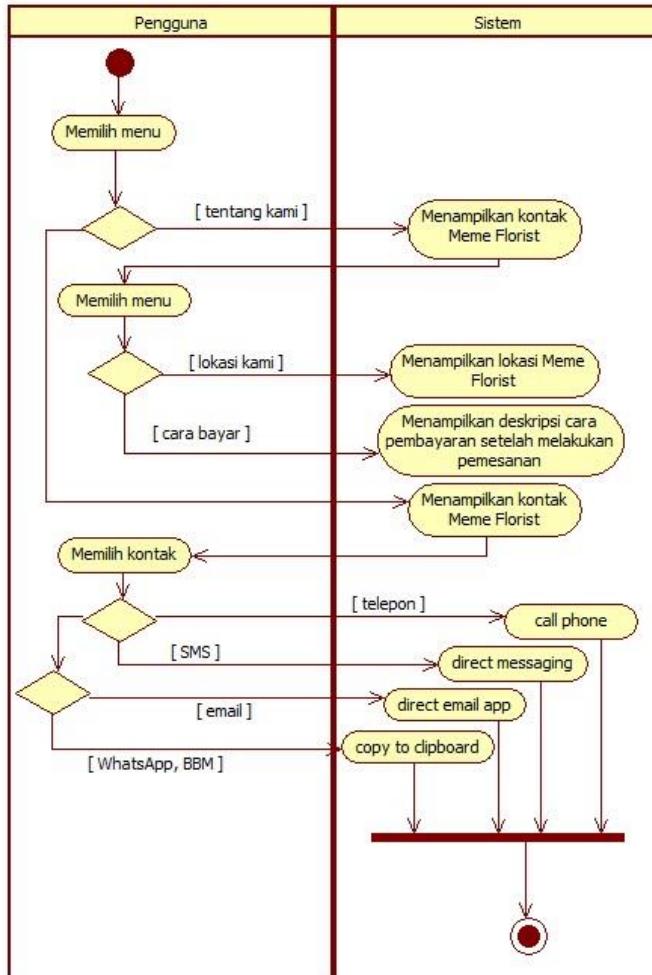
[Spesifikasi Pengembangan yang terkait: Subbab 4.1.1 poin o]

Untuk melengkapi semua itu, dokumen ini dilengkapi diagram aktivitas untuk mengetahui bagaimana alur untuk tiap kasus penggunaan. Diagram aktivitas untuk kasus melihat katalog bunga ditunjukkan oleh Gambar 4.2, diagram aktivitas untuk melihat informasi Meme Florist ditunjukkan oleh Gambar 4.3, diagram aktivitas untuk kasus mengelola keranjang belanja ditunjukkan oleh **Error! Reference source not found.**, diagram aktivitas untuk kasus mengelola profil pengguna ditunjukkan oleh Gambar 4.5, diagram aktivitas untuk kasus melihat histori pemesanan ditunjukkan oleh Gambar 4.6, diagram aktivitas untuk kasus melihat histori notifikasi ditunjukkan oleh Gambar 4.7, diagram aktivitas untuk kasus registrasi ditunjukkan oleh Gambar 4.8, diagram aktivitas untuk kasus meminta *reset password*

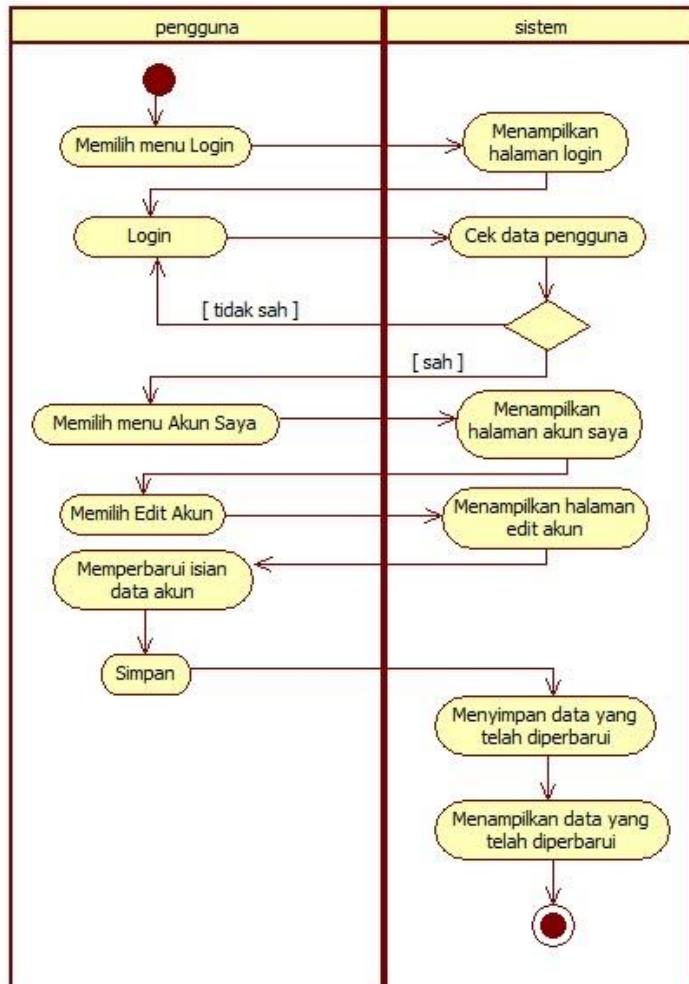
ditunjukkan oleh Gambar 4.9, diagram aktivitas untuk kasus memesan bunga ditunjukkan oleh Gambar 4.10, diagram aktivitas untuk kasus menerima *push notification* ditunjukkan oleh Gambar 4.11, dan diagram aktivitas untuk kasus melakukan mengatur penerimaan notifikasi ditunjukkan oleh Gambar 4.12.



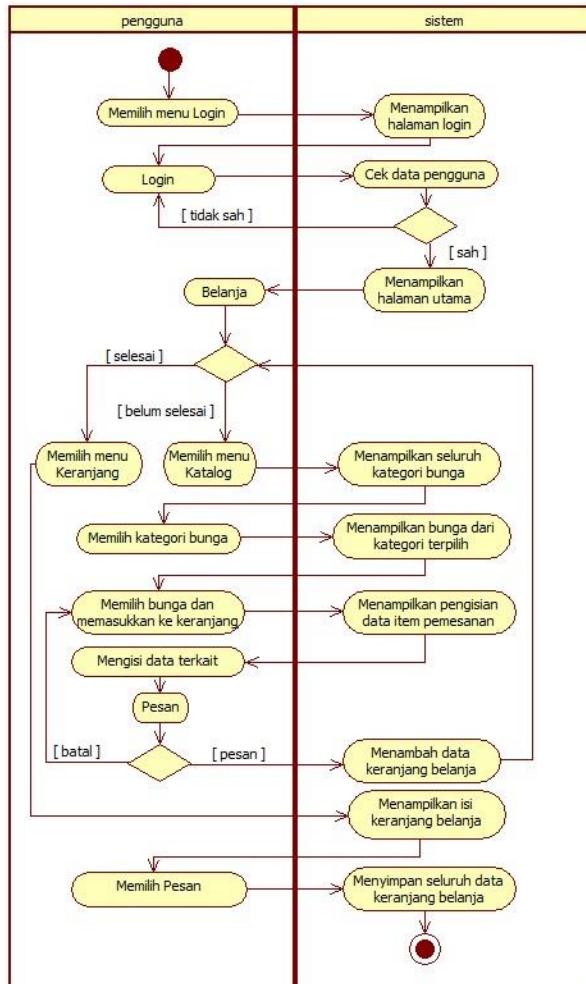
Gambar 4.2 Diagram Aktivitas untuk Kasus Melihat Katalog Bunga



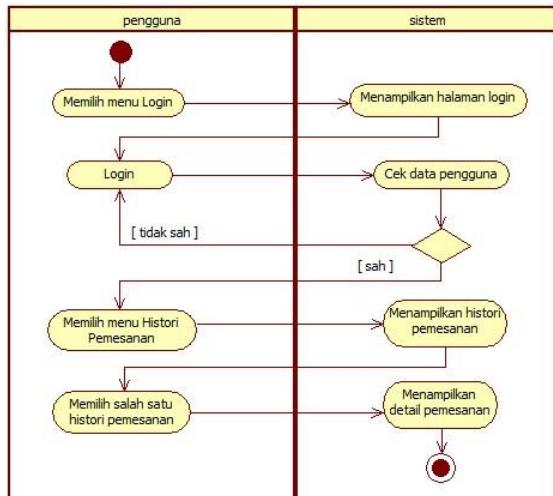
Gambar 4.3 Diagram Aktivitas untuk Melihat Informasi tentang Meme Florist



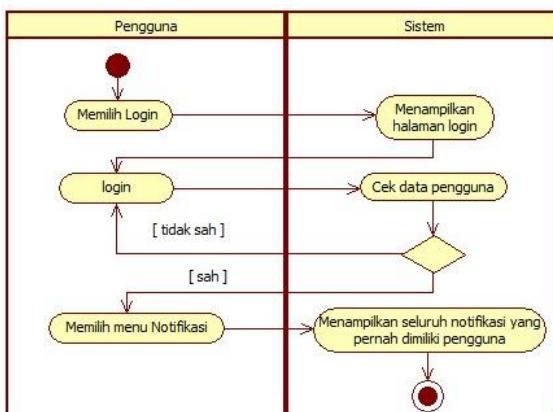
Gambar 4. 4 Diagram Aktivitas untuk Mengelola Keranjang Belanja



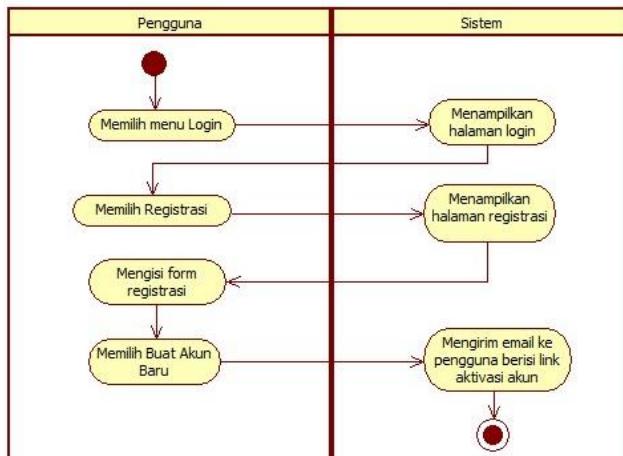
Gambar 4.5 Diagram Aktivitas untuk Kasus Mengelola Profil Pengguna



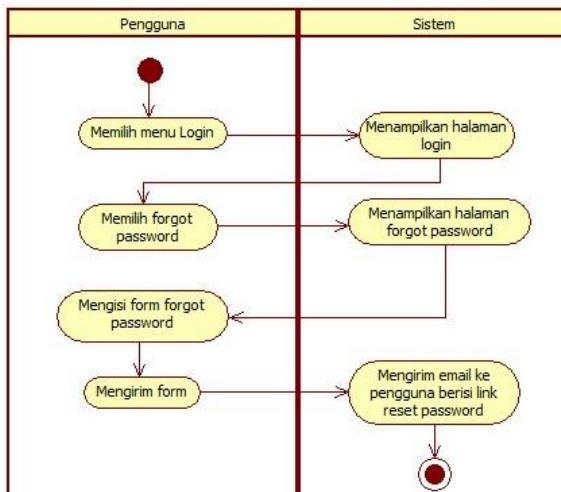
Gambar 4.6 Diagram Aktivitas untuk Kasus Melihat Histori Pemesanan



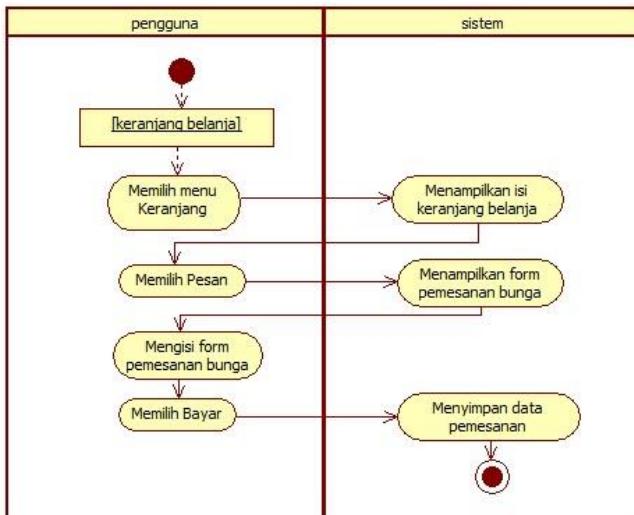
Gambar 4.7 Diagram Aktivitas untuk Kasus Melihat Histori Notifikasi



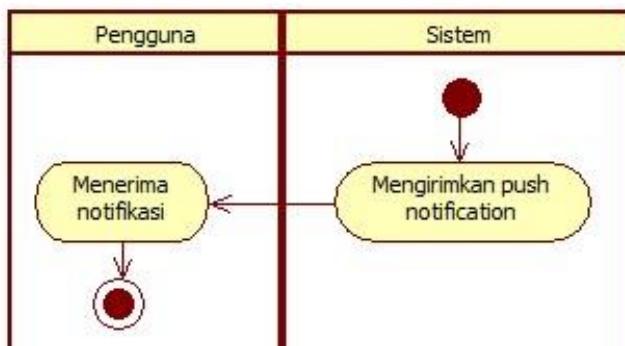
Gambar 4.8 Diagram Aktivitas untuk Kasus Registrasi



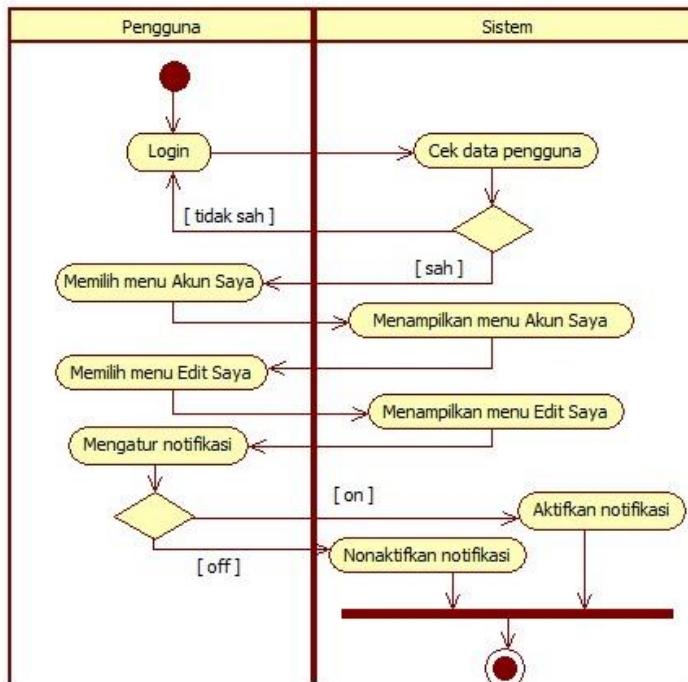
Gambar 4.9 Diagram Aktivitas untuk Kasus Meminta Reset Password



Gambar 4.10 Diagram Aktivitas untuk Kasus Memesan Bunga



Gambar 4.11 Diagram Aktivitas untuk Kasus Menerima Push Notification



Gambar 4.12 Diagram Aktivitas untuk Kasus Mengatur Penerimaan Notifikasi

- **Perancangan Antarmuka**

- a. **Antarmuka Halaman *Splashscreen***

Untuk rancangan antarmuka halaman *splashscreen* yang ditunjukkan pada **Error! Reference source not found.**, berisi latar belakang dengan warna dominan aplikasi dan logo aplikasi. Logo aplikasi pada *splashscreen* ditampilkan dengan menggunakan kontrol *ImageView* dan animasi *fade in* dan *fade out*.

b. Antarmuka *Landing Menu* untuk Pengguna yang Tidak login

Untuk rancangan antarmuka halaman *landing menu* untuk pengguna yang tidak *login*, berisi menu katalog, tentang kami, dan hubungi kami. *Landing menu* yang digunakan adalah jenis *menu drawer* yang berbentuk kelas *Fragment*, yaitu muncul dari sebelah kanan atau kiri aplikasi, sedangkan logo aplikasi digunakan untuk menampilkan *menu drawer*. Seperti yang ditunjukkan pada **Error! Reference source not found.**, menu pada *drawer* masing-masingnya adalah berupa *icon image*. Saat salah satu menu dipilih, antarmuka akan berpindah dari *menu drawer* ke antarmuka menu yang dipilih tersebut, hal ini menggunakan kontrol perpindahan antar



Gambar 4.13 Rancangan antarmuka *splashscreen*



Gambar 4.14 Rancangan antarmuka *landing menu* pengguna tidak login

kelas *Fragment*.

c. Antarmuka Halaman Katalog untuk Pengguna yang Tidak Login

Untuk rancangan antarmuka halaman katalog untuk pengguna yang tidak *login* berisi seluruh kategori dan detail bunga pada masing-masing kategori. Seperti yang ditunjukkan pada **Error! Reference source not found.**, kategori bunga ditampilkan dengan kontrol *GridView*, sedangkan seperti yang ditunjukkan pada **Error! Reference source not found.** untuk detail bunga pada anggota masing-masing kategori ditampilkan dengan kontrol *pagerview*. Antarmuka kategori bunga adalah berupa kelas *Fragment*, sedangkan untuk antarmuka detail bunga menggunakan kelas *ActionBarActivity*. Saat salah satu kategori dipilih, antarmuka akan berpindah dari antarmuka kategori ke antarmuka detail bunga berdasarkan kategori yang dipilih tersebut, hal ini menggunakan kontrol perpindahan dari kelas *Fragment* ke kelas *ActionBarActivity*.

Pada ujung atas kanan antarmuka pada **Error! Reference source not found.** terdapat tombol *sort* harga yang dapat digunakan untuk mengurutkan harga bunga dari yang paling mahal dan mengurutkan bunga dari yang paling murah. Tombol *sort* ini menggunakan kontrol *ImageButton*.

d. Antarmuka Halaman Informasi tentang Meme Florist

Untuk rancangan antarmuka halaman informasi tentang Meme Florist berisi deskripsi mengenai Meme Florist, tombol Lokasi Kami yang akan menampilkan peta lokasi meme florist dengan Google Maps, dan tombol Cara Bayar. Seperti yang ditunjukkan pada **Error! Reference source not found.** deskripsi menggunakan kontrol *textview*, sedangkan tombol Lokasi Kami dan Bayar Sekarang menggunakan kontrol *ImageButton*. Antarmuka tentang kami merupakan kelas *Fragment*, sehingga tombol Lokasi Kami dan Bayar Sekarang merupakan kelas *Activity*, sehingga jika salah satu dari kedua tombol tersebut ditekan, maka akan terjadi perpindahan antarmuka dari kelas *Fragment* ke kelas *Activity*.



Gambar 4. 16 Rancangan antarmuka katalog pada kategori bunga



Gambar 4. 15 Rancangan antarmuka katalog pada detail bunga

e. Antarmuka Halaman Informasi tentang Kontak Meme Florist

Untuk rancangan antarmuka halaman informasi tentang kontak Meme Florist berisi seluruh kontak Meme Florist dan tombol *Save to Contact* untuk menyimpan seluruh kontak tersebut ke dalam *contact device* pengguna. Seperti yang ditunjukkan pada **Error! Reference source not found.**, masing-masing kontak terdapat kontrol *ImageButton* untuk diarahkan langsung ke aplikasi-aplikasi Android seperti *call phone*, *messaging*, WhatsApp, BlackBerry Messenger, dan *email app*. Begitu pula pada tombol *Save to Contact* yang menggunakan kontrol *imagebutton*, ketika ditekan, antarmuka langsung berpindah ke *contact app* untuk menyimpan kontak ke dalam *device*. Antarmuka halaman ini menggunakan kontrol *listview* dalam menampilkan seluruh kontak Meme Florist.

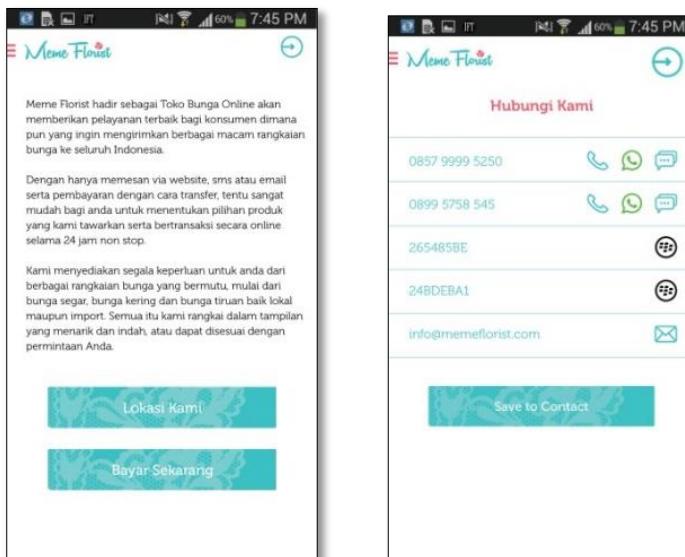
f. Antarmuka Halaman Lokasi Kami

Untuk rancangan antarmuka halaman informasi tentang Lokasi Kami berisi lokasi Meme Florist yang ditampilkan oleh Google Maps. Seperti yang ditunjukkan pada **Error! Reference source not found.**, antarmuka ini ditujukan untuk pengguna lebih fokus terhadap lokasi Meme Florist dalam peta sehingga diberikan *marker* pada lokasi Meme Florist berdasarkan *latitude* dan *longitude* beserta

penggunaan *zoom in camera* pada *maps* yang ditampilkan.

g. Antarmuka Halaman Cara Pembayaran

Untuk rancangan antarmuka halaman cara pembayaran berisi nomor rekening yang dimiliki Meme Florist. Hal ini ditujukan untuk pengguna yang telah melakukan pemesanan melalui aplikasi Meme Florist dapat mengirimkan biaya pemesanan bunga melalui rekening yang telah tersedia. Seperti yang ditunjukkan pada **Error! Reference source not found.**, antarmuka nomor rekening Meme Florist menggunakan kontrol



listview.



Gambar 4.17
Rancangan antarmuka halaman informasi tentang Meme Florist



Gambar 4.19
Rancangan antarmuka cara pembayaran

h. Antarmuka *Landing Menu* untuk Pengguna yang Telah *Login*

Untuk rancangan antarmuka halaman *landing menu* untuk pengguna yang telah *login*, berisi menu katalog, histori pemesanan, akun saya, tentang kami, histori notifikasi, hubungi kami, dan keranjang. *Landing menu* yang digunakan adalah jenis *menu drawer* yang

berbentuk kelas *Fragment*. Seperti yang ditunjukkan pada **Error! Reference source not found.**, menu pada *drawer* masing-masingnya adalah berupa *icon image*. Saat salah satu menu dipilih, antarmuka akan berpindah dari *menu drawer* ke antarmuka menu yang dipilih tersebut, hal ini menggunakan kontrol perpindahan antar kelas *Fragment*.

i. **Antarmuka Halaman Katalog untuk Pengguna yang Telah Login**

Untuk rancangan antarmuka halaman katalog untuk pengguna yang telah *login* berisi seluruh kategori dan detail bunga pada masing-masing kategori. Seperti yang ditunjukkan pada **Error! Reference source not found.** kategori bunga ditampilkan dengan kontrol *GridView*, sedangkan seperti yang ditunjukkan pada **Error! Reference source not found.** untuk detail bunga pada anggota masing-masing kategori ditampilkan dengan kontrol *pagerview* beserta tombol masukkan ke keranjang. Antarmuka kategori bunga adalah berupa kelas *Fragment*, sedangkan untuk antarmuka detail bunga menggunakan kelas *ActionBarActivity*. Saat salah satu kategori dipilih, antarmuka akan berpindah dari antarmuka kategori ke antarmuka detail bunga berdasarkan kategori yang dipilih tersebut, hal ini menggunakan kontrol perpindahan dari kelas *Fragment* ke kelas *ActionBarActivity*. Pada ujung atas kanan antarmuka pada **Error! Reference**

source not found. terdapat tombol *sort* harga yang dapat digunakan untuk mengurutkan harga bunga dari yang paling mahal dan mengurutkan bunga dari yang paling murah. Tombol *sort* ini menggunakan kontrol *ImageButton*. Untuk **Error! Reference source not found.** yang ada dibawah ini adalah antarmuka ketika tombol Masukkan ke Keranjang ditekan. Antarmuka tersebut menampilkan kotak dialog berupa jumlah pemesanan bunga dan isi pesan untuk bunga yang dipilih dengan menggunakan kontrol *AlertDialog*.

Ketika seluruh *field* telah terisi, tombol *Cancel* digunakan untuk membatalkan pemesanan, sedangkan tombol pesan digunakan untuk menyimpan data bunga yang dipesan ke dalam keranjang belanja.

j. Antarmuka Halaman Akun Saya

Untuk rancangan antarmuka halaman akun saya menampilkan nama, *email*, nomor telepon, alamat, dan pengaturan notifikasi untuk pesan dari Meme Florist. Seperti yang ditunjukkan pada **Error! Reference source not found.** profil pengguna ditampilkan menggunakan kontrol *TextView*, sedangkan untuk tombol Edit Data ditampilkan menggunakan kontrol *ImageButton*.

k. Antarmuka Halaman Edit Akun Saya

Untuk rancangan antarmuka halaman edit akun saya menampilkan data profil pengguna yang dapat diperbarui seperti nama, *email*, nomor

telepon, alamat, dan pengaturan notifikasi untuk pesan dari Meme Florist. Seperti yang ditunjukkan pada **Error! Reference source not found.** profil pengguna ditampilkan menggunakan kontrol *TextView* dan *EditText*, sedangkan untuk tombol simpan data ditampilkan menggunakan kontrol *ImageButton*.



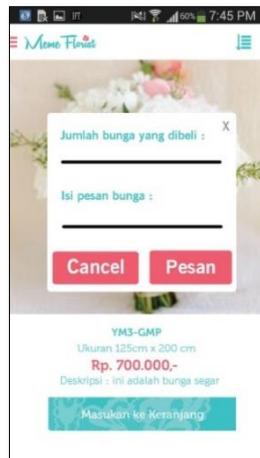
Gambar 4.21 Rancangan antarmuka *landing menu* untuk pengguna yang telah *login*



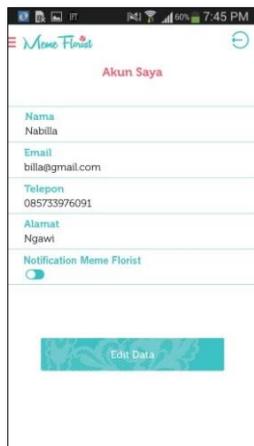
Gambar 4. 22 Rancangan antarmuka halaman kategori bunga



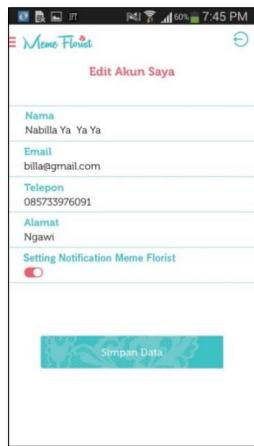
Gambar 4. 24 Rancangan antarmuka halaman detail bunga



Gambar 4. 23 Rancangan antarmuka halaman masukkan ke keranjang



Gambar 4.25
Rancangan antarmuka halaman akun saya



Gambar 4.26
Rancangan antarmuka halaman edit akun saya

I. Antarmuka Halaman Histori Pemesanan

Seperti yang ditunjukkan pada **Error! Reference source not found.** rancangan antarmuka halaman histori pemesanan menampilkan *list* pemesanan yang pernah dilakukan oleh pengguna. *List* histori pemesanan ini ditampilkan dengan kontrol *ListView*, ketika salah satu *list* tersebut ditekan, maka antarmuka akan berpindah ke detail histori pemesanan seperti yang ditunjukkan pada **Error! Reference source not found..** Detail histori pemesanan ditampilkan dengan kontrol *ListView*, sedangkan untuk gambar bunga pada masing-masing *item* menggunakan kontrol *ImageView* dan detail informasi bunga menggunakan kontrol *TextView*.



Gambar 4. 28 Rancangan antarmuka halaman detail histori pemesanan



Gambar 4. 27 Rancangan antarmuka halaman histori pemesanan

m. Antarmuka Histori Notifikasi

Untuk rancangan antarmuka halaman histori notifikasi menampilkan pesan notifikasi dari Meme Florist mengenai promosi produk bunga. Seperti yang ditunjukkan pada **Error! Reference source not found.**, menampilkan *list* notifikasi yang pernah didapatkan oleh pengguna. *List* histori pemesanan ini ditampilkan dengan kontrol *ListView*, logo Meme Florist ditampilkan dengan kontrol *ImageView*, dan teks pesan notifikasi menggunakan kontrol *TextView*.

n. Antarmuka Halaman Keranjang Belanja

Seperti yang ditunjukkan pada **Error! Reference source not found.**, rancangan antarmuka halaman keranjang belanja menampilkan *list* bunga yang telah masuk ke keranjang belanja. *List* detail bunga gambar bunga ditampilkan dengan kontrol *ImageView*, informasi detil bunga menggunakan kontrol *TextView*, dan tombol tambah dan kurang kuantitas pemesanan bunga ditampilkan dengan kontrol *ImageButton*. Pada antarmuka ini juga terdapat informasi total harga yang harus dibayar oleh pengguna atas pemesanan bunga yang telah dilakukan dan tombol Pesan untuk melakukan proses pemesanan bunga lebih lanjut. Informasi total harga tersebut ditampilkan dengan kontrol *TextView* dan tombol pesan ditampilkan dengan kontrol *ImageButton*.

o. Antarmuka Halaman Login

Untuk rancangan antarmuka halaman *login* menampilkan *field* untuk melakukan proses *login*, menampilkan tombol *login with Google*, tombol *register* dan tombol *forgot password*. Pada antarmuka ini, proses *login* dapat dilakukan dengan dua proses yaitu dengan menggunakan akun Meme Florist yang telah terdaftar dengan memasukkan *email* dan *password* atau juga dapat menggunakan *email* saja melalui *login* dengan Google. Seperti yang ditunjukkan pada **Error! Reference source not found.**, logo Meme Florist ditampilkan dengan kontrol *ImageView*, tombol

login with Google ditampilkan dengan kontrol *ImageButton*, *field email* dan *password* ditampilkan dengan kontrol *EditText*, tombol *login* ditampilkan dengan kontrol *ImageButton*, sedangkan tombol *ForgotPassword* ditampilkan dengan kontrol *TextView*.

p. Antarmuka Halaman Registrasi

Untuk rancangan antarmuka halaman Registrasi menampilkan *field* untuk melakukan proses pendaftaran akun Meme Florist dan menampilkan tombol Buat Akun Baru. Seperti yang ditunjukkan pada **Error! Reference source not found.**, *field* pengisian data diri untuk registrasi ditampilkan dengan kontrol *TextView* dan *EditText*, sedangkan tombol Buat Akun Baru ditampilkan dengan kontrol *ImageButton*.

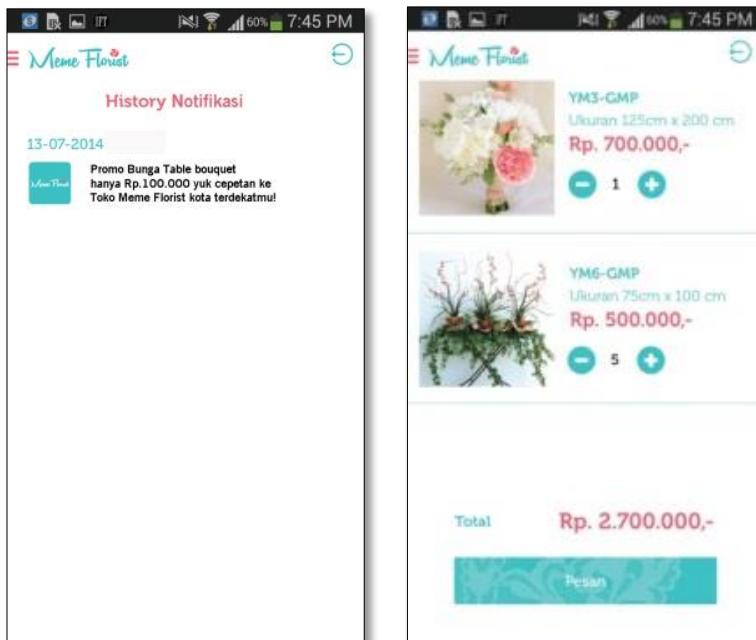
q. Antarmuka Halaman *Forgot Password*

Untuk rancangan antarmuka halaman *forgot password* menampilkan *field* untuk melakukan proses konfirmasi lupa *password* dengan memasukkan *email* pengguna. Seperti yang ditunjukkan pada **Error! Reference source not found.**, *field* pengisian *email* menggunakan kontrol *TextView* dan *EditText*, sedangkan tombol Kirim *Email* menggunakan kontrol *ImageButton*.

r. Antarmuka Halaman Pemesanan Bunga

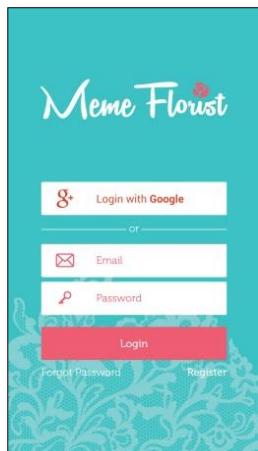
Untuk rancangan antarmuka halaman pemesanan bunga menampilkan *field* untuk

melakukan proses pengisian data untuk tujuan pengiriman bunga yang telah dipesan. Seperti yang ditunjukkan pada **Error! Reference source not found.**, field pengisian data tujuan pengiriman ditampilkan menggunakan kontrol *TextView* dan *EditText*, sedangkan tombol bayar sekarang menggunakan kontrol *ImageButton*.

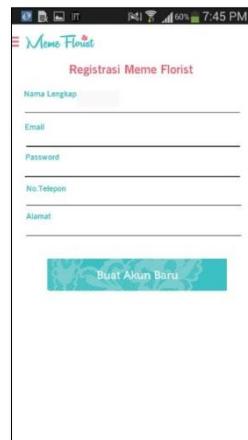


Gambar 4.30
Rancangan antarmuka
halaman histori
notifikasi

Gambar 4. 29 Rancangan
antarmuka halaman



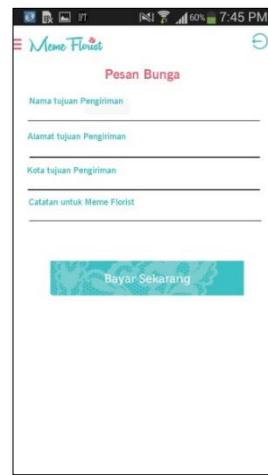
Gambar 4.32 Rancangan antarmuka halaman *login*



Gambar 4.33 Rancangan antarmuka halaman registrasi



Gambar 4.31 Rancangan antarmuka halaman *forgot password*



Gambar 4.34 Rancangan antarmuka halaman pesan bunga

4.2. Analisis dan Perancangan Aplikasi Web

Pada subab ini akan dijelaskan mengenai analisis dan perancangan aplikasi Meme Florist pada perangkat *mobile*.

4.2.1 Analisis Kebutuhan

Secara umum spesifikasi kebutuhan perangkat lunak yang akan dikembangkan dalam aplikasi *web* administrator yang seterusnya akan diberi nama aplikasi *web* Meme Florist ini adalah sebagai berikut:

a. Mengelola data katalog pada aplikasi *web* Meme Florist

Data katalog yang ditampilkan pada aplikasi Meme Florist dikelola melalui aplikasi *web* Meme Florist. Tujuannya agar administrator dapat menambahkan, mengubah, maupun menghapus data katalog.

b. Mengelola data profil dan kontak toko Meme Florist pada aplikasi *web* Meme Florist

Profil toko Meme Florist yang ditampilkan pada aplikasi Meme Florist dapat dikelola melalui aplikasi *web* Meme Florist. Adanya pengelolaan data profil toko ini bertujuan agar pemilik toko atau administrator dapat dengan mudah mengubah maupun menambah data yang diperlukan seperti rekening bank, alamat toko, alamat *email*, nomor telepon, dan lain sebagainya.

c. Menambahkan sistem konfirmasi pengiriman pemesanan bunga

Pada setiap pemesanan yang telah dilakukan oleh pelanggan melalui aplikasi Meme Florist akan diproses

oleh administrator terlebih dahulu melalui aplikasi *web* Meme Florist. Jika pemesanan telah selesai diproses dan barang berhasil dikirim ke tempat tujuan, maka *administrator* akan memberikan konfirmasi pengiriman berupa *push notification* kepada pemesan terkait. Tujuan dengan penambahan sistem konfirmasi ini adalah agar pemesan mengetahui bahwa bunga yang ia pesan telah sampai di tujuan dan sebagai penanda bagi pemilik toko bahwa pemesanan telah selesai diproses.

- d. Menampilkan daftar pemesanan bunga pelanggan**
Setiap transaksi yang telah dilakukan oleh pelanggan disimpan ke dalam Datastore. Kemudian, administrator dapat melihat daftar pemesanan pelanggan yang masuk agar bisa diproses lebih lanjut. Daftar pemesanan pelanggan disajikan dalam bentuk tabel agar memudahkan administrator dalam mengolah pemesanan.
- e. Mengirimkan pesan pemberitahuan kepada pelanggan**
Administrator dapat mengirimkan pesan pemberitahuan kepada pelanggannya berupa *push notification* dan dapat diterima pelanggan di masing-masing *device*-nya. Tujuan dari pengiriman pesan pemberitahuan ini adalah agar Meme Florist dapat senantiasa berkomunikasi dengan pelanggan-pelanggannya seperti ketika Meme Florist sedang mengadakan promosi.

f. Menambahkan sistem aktivasi akun pelanggan

Ketika pelanggan mendaftar, sistem belum bisa secara langsung mendeteksi bahwa alamat *email* yang digunakan oleh pelanggan adalah valid. Sehingga, diperlukan adanya sistem aktivasi akun pelanggan melalui *email* yang bertujuan untuk memastikan bahwa *email* yang digunakan oleh pelanggan adalah valid dan tersedia.

g. Menambahkan sistem *forgot password* akun pelanggan

Kadangkala pelanggan bisa lupa terhadap *password* yang ia gunakan untuk bisa *login* ke dalam aplikasi Meme Florist. Oleh karena itu, dengan penambahan sistem *forgot password* ini bertujuan agar pelanggan dapat mengganti *password* yang lama dengan *password* yang baru melalui *link* yang dikirimkan ke *email* pelanggan terdaftar.

4.2.2 Definisi Umum Aplikasi

Secara umum, aplikasi *web* ini ditujukan untuk pengelola ataupun pemilik toko Meme Florist yang selanjutnya disebut sebagai administrator Meme Florist. Tujuan pembuatan aplikasi *web* ini adalah untuk memudahkan administrator dalam mengelola data katalog, profil toko, dan pemesanan pelanggan, serta mengirimkan promosi kepada pelanggan.

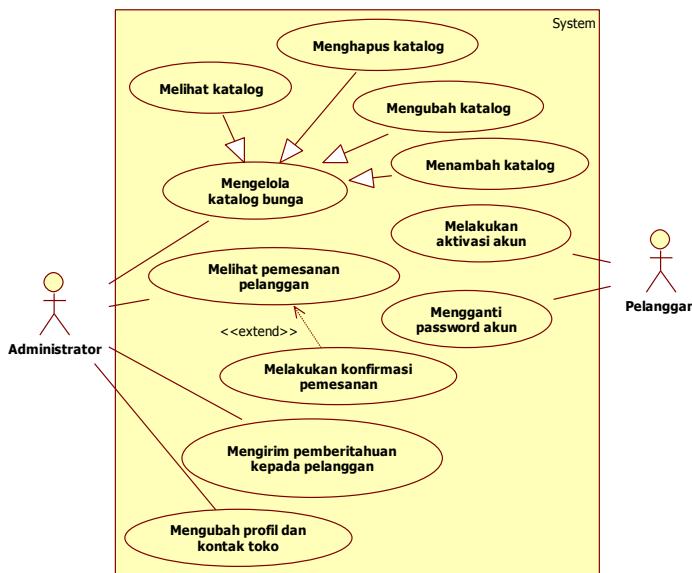
Data katalog bunga yang dipesan oleh pelanggan melalui aplikasi Meme Florist dikelola melalui aplikasi *web*. Setelah pelanggan mengirimkan data pemesanannya, maka selanjutnya administrator dapat mengeceknya di aplikasi *web*.

dan memprosesnya. Jika barang sudah dikirimkan ke tempat tujuan, maka administrator akan memberikan konfirmasi pengiriman pemesanan melalui *push notification* dan *email* kepada pemesan terkait. Selain itu, pelanggan dapat melihat profil dan kontak dari toko dengan data yang dikelola melalui *web* serta administrator dapat mengirimkan promosi kepada pelanggan melalui *push notification*.

4.2.3 Perancangan Fungsionalitas Aplikasi

- **Perancangan Fungsionalitas Aplikasi**

Berdasarkan uraian di atas, gambaran spesifikasi kebutuhan dapat digambarkan dengan diagram kasus penggunaan berikut pada Gambar 4.35.



Gambar 4.35 Kasus Penggunaan Aplikasi Mobile Web Meme Florist

- **Cara Kerja Aplikasi**

Aplikasi *web* ini memiliki keterkaitan dengan aplikasi *mobile* Meme Florist dalam proses bisnis pemesanan bunga Meme Florist. Berikut adalah cara kerja aplikasi *web* Meme Florist:

1. **Login Administrator Meme Florist**

Sebagai upaya keamanan penggunaan aplikasi *web* Meme Florist, administrator harus melakukan *login* terlebih dahulu untuk dapat menggunakan fitur-fitur dari aplikasi *web* Meme Florist.

2. **Melihat daftar katalog bunga**

Administrator dapat melihat daftar katalog bunga yang telah tersimpan di dalam sistem. Untuk mempermudah administrator dalam menemukan dan memilih bunga berdasarkan kategorinya, administrator dapat menggunakan fitur *filter* kategori.

3. **Menambah data katalog bunga**

Katalog bunga yang ditampilkan di aplikasi Meme Florist disimpan oleh sistem aplikasi *web*. Untuk menambah daftar katalog bunga yang tersedia, administrator dapat menambahkannya melalui aplikasi *web* ini.

4. **Mengubah data katalog bunga**

Administrator dapat mengubah data katalog bunga yang ditentukan jika suatu ketika ada perubahan harga, foto bunga, ataupun yang lain melalui

aplikasi web ini dengan memilih terlebih dahulu katalog yang akan diubah datanya.

5. Menghapus data katalog bunga

Jika terdapat data katalog bunga yang akan dihapus dari penyimpanan, melalui aplikasi *web* ini administrator dapat menghapusnya dengan terlebih dahulu memilih katalog yang akan dihapus.

6. Melihat daftar pemesanan pelanggan

Data pemesanan yang telah dikirimkan oleh pelanggan melalui aplikasi Meme Florist akan tersimpan ke dalam sistem aplikasi *web*, kemudian administrator dapat melihat kembali daftar pemesanan pelanggan-pelanggan yang telah masuk. Sehingga setelah ditampilkan daftar pemesanannya, administrator dapat melanjutkan proses transaksi dengan memproses pemesanan, yakni mengirimkan bunga pesanan yang diinginkan pelanggan ke tujuan.

7. Melakukan konfirmasi pemesanan bunga

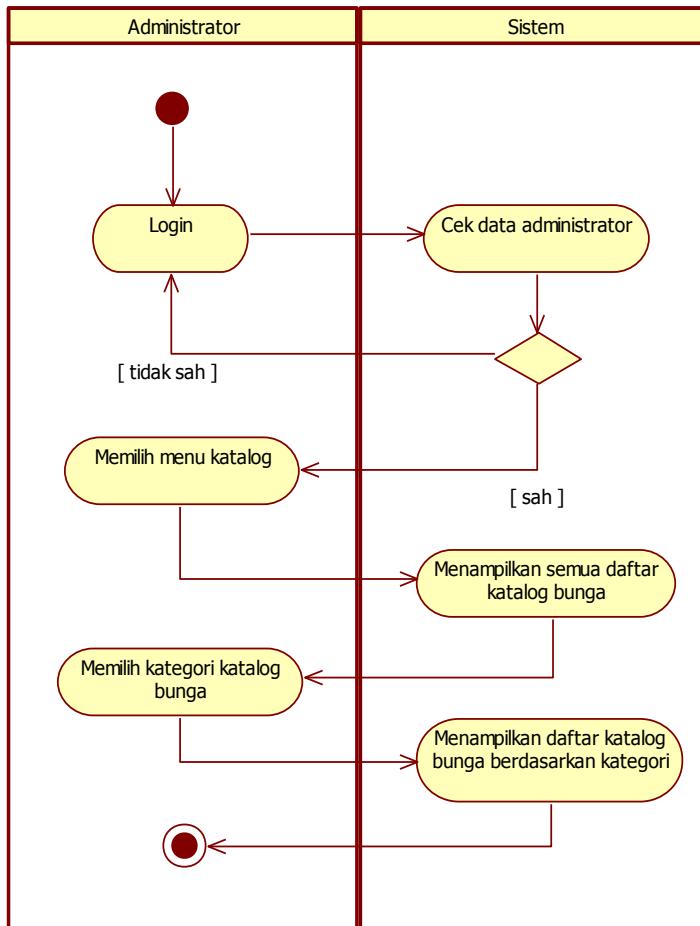
Jika pemesanan telah diproses dan barang telah selesai dikirim, maka transaksi selesai. Selanjutnya administrator akan melakukan konfirmasi kepada pelanggan yang bersangkutan bahwa barang telah dikirim ke tujuan. Konfirmasi yang dikirimkan ke pelanggan berupa *push notification* yang akan muncul di aplikasi Meme Florist dan *email*.

- 8. Mengirim pemberitahuan kepada pelanggan**
Administrator dapat mengirimkan pemberitahuan kepada pelanggan melalui aplikasi *web* ini. Pemberitahuan atau notifikasi yang dikirimkan berupa *push notification*. Pesan ini bisa berupa promosi ataupun yang lain.
- 9. Mengubah profil dan kontak toko Meme Florist**
Profil dan kontak toko yang tertera di aplikasi Meme Florist dapat diatur oleh administrator melalui aplikasi *web* ini. Profil toko berisi sekilas tentang toko bunga Meme Florist, sedangkan kontak toko berisi data-data yang dapat dihubungi oleh pelanggan untuk berkomunikasi dengan toko Meme Florist, bisa berupa nomor rekening bank untuk pembayaran, alamat *email*, nomor telepon, dan lain sebagainya.
- 10. Melakukan aktivasi akun pelanggan**
Setelah pelanggan melakukan pendaftaran melalui aplikasi Meme Florist maka akun pelanggan perlu diaktifasi terlebih dahulu untuk memastikan bahwa *email* yang digunakan oleh pelanggan dalam registrasi adalah valid. Pelanggan akan mendapatkan sebuah *link* yang akan mengarahkan ke halaman aktivasi yang disediakan oleh aplikasi *web*.

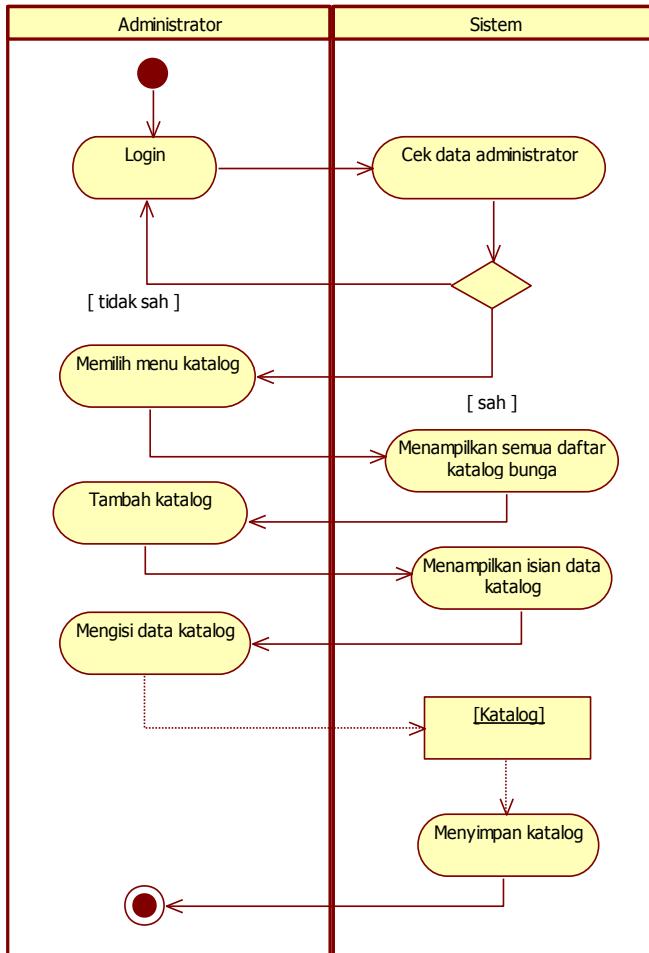
11. Mengubah *password* akun pelanggan

Pelanggan terkadang lupa dengan *password* akunnya. Oleh karena itu, agar pelanggan tersebut dapat menggunakan akunnya, maka aplikasi *web* memberikan fasilitas untuk melakukan *reset password*.

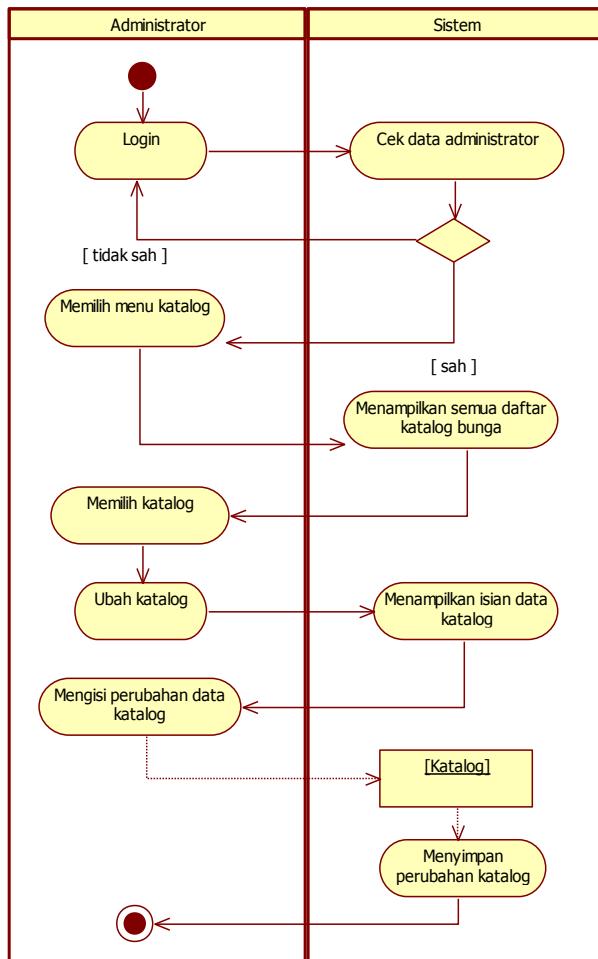
Untuk melengkapi semua kasus penggunaan di atas, maka dokumen ini telah dilengkapi dengan diagram aktivitas untuk mengetahui bagaimana alur untuk setiap kasus penggunaannya. Yaitu, diagram aktivitas untuk kasus melihat daftar katalog bunga yang ditunjukkan oleh Gambar 4.36, diagram aktivitas untuk kasus menambah data katalog bunga yang ditunjukkan oleh Gambar 4.37, diagram aktivitas untuk kasus mengubah data katalog bunga yang ditunjukkan oleh Gambar 4.38, diagram aktivitas untuk kasus menghapus data katalog bunga yang ditunjukkan oleh Gambar 4.39, diagram aktivitas untuk kasus melihat daftar pemesanan pelanggan yang ditunjukkan oleh Gambar 4.40, diagram aktivitas untuk kasus melakukan konfirmasi pemesanan bunga yang ditunjukkan oleh Gambar 4.41, diagram aktivitas untuk kasus mengirim pemberitahuan kepada pelanggan yang ditunjukkan oleh Gambar 4.42, diagram aktivitas untuk kasus mengubah profil dan kontak toko Meme Florist yang ditunjukkan oleh Gambar 4.43, diagram aktivitas untuk kasus melakukan aktivasi akun pelanggan yang ditunjukkan oleh Gambar 4.44, dan diagram aktivitas untuk kasus mengubah *password* akun pelanggan yang ditunjukkan oleh Gambar 4.45.



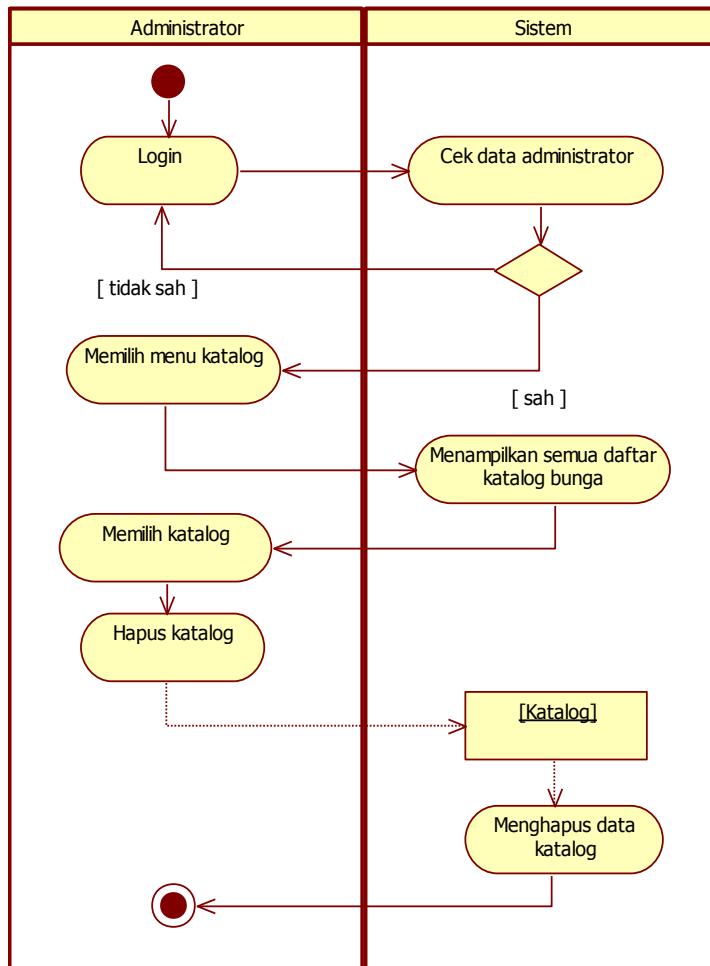
Gambar 4.36 Diagram Aktivitas untuk Kasus Melihat Daftar Katalog Bunga



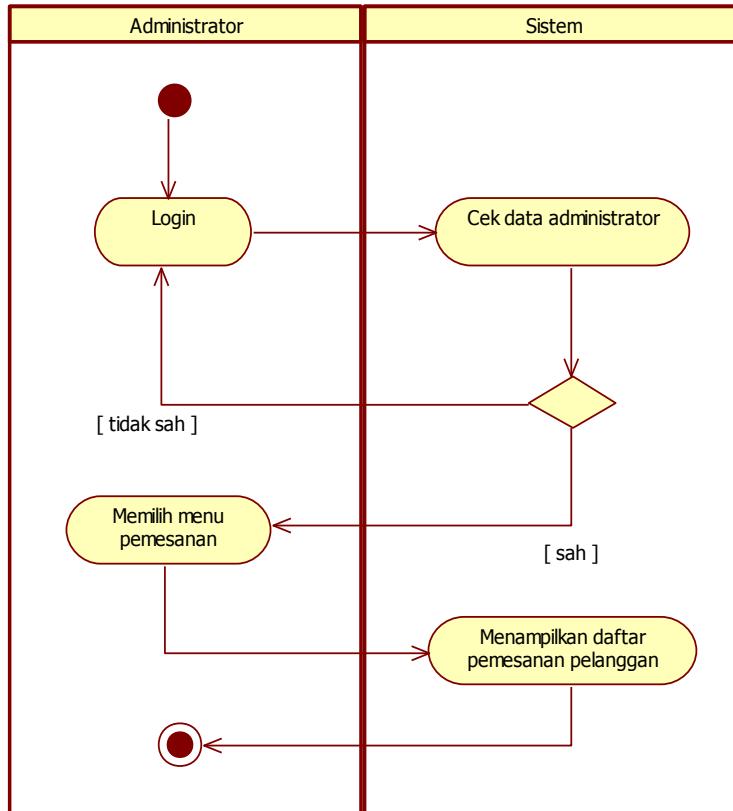
Gambar 4.37 Diagram Aktivitas untuk Kasus Menambah Data Katalog Bunga



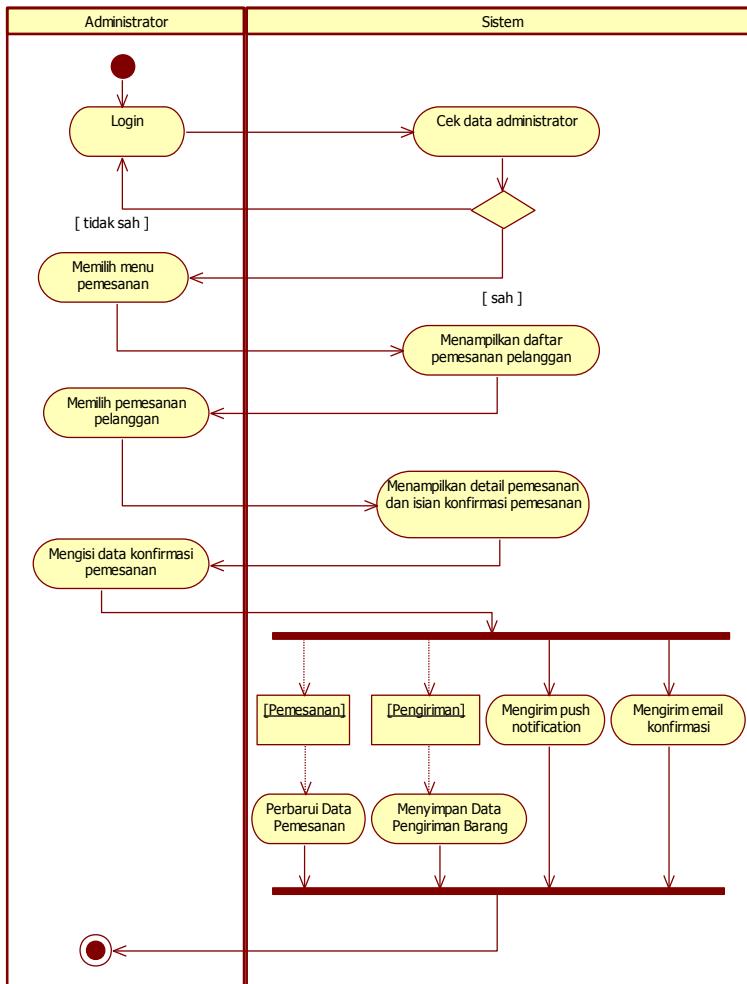
Gambar 4.38 Diagram Aktivitas untuk Kasus Mengubah Data Katalog Bunga



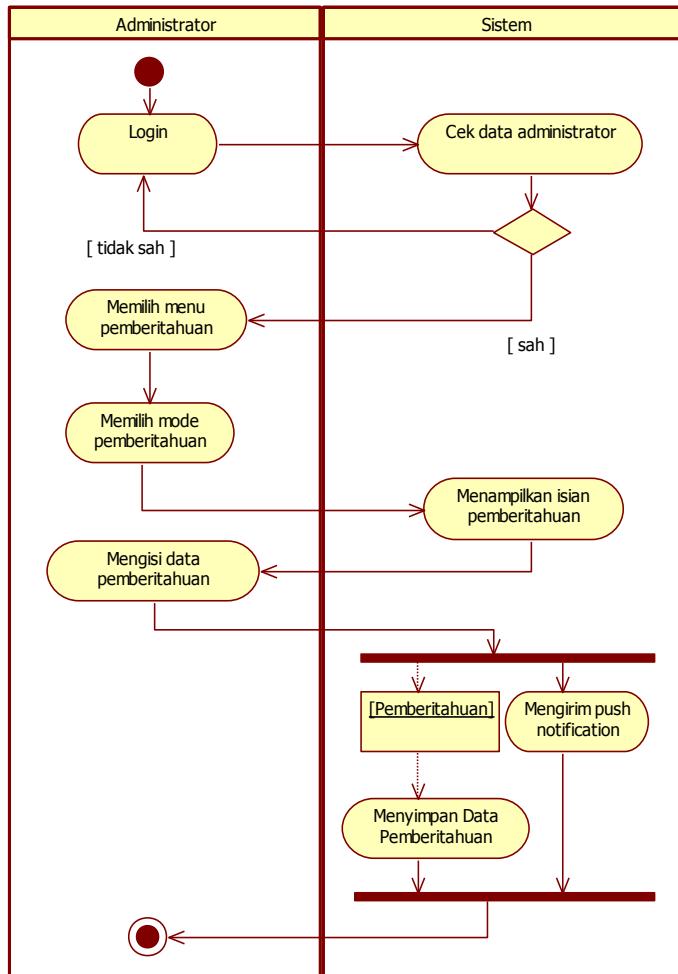
Gambar 4.39 Diagram Aktivitas untuk Kasus Menghapus Data Katalog Bunga



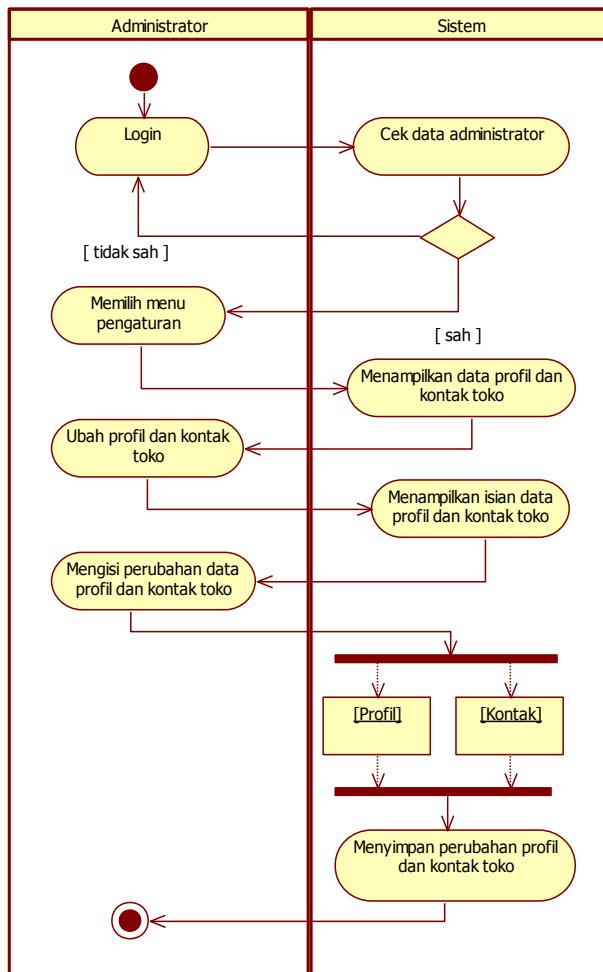
Gambar 4.40 Diagram Aktivitas untuk Kasus Melihat Daftar Pemesanan Pelanggan



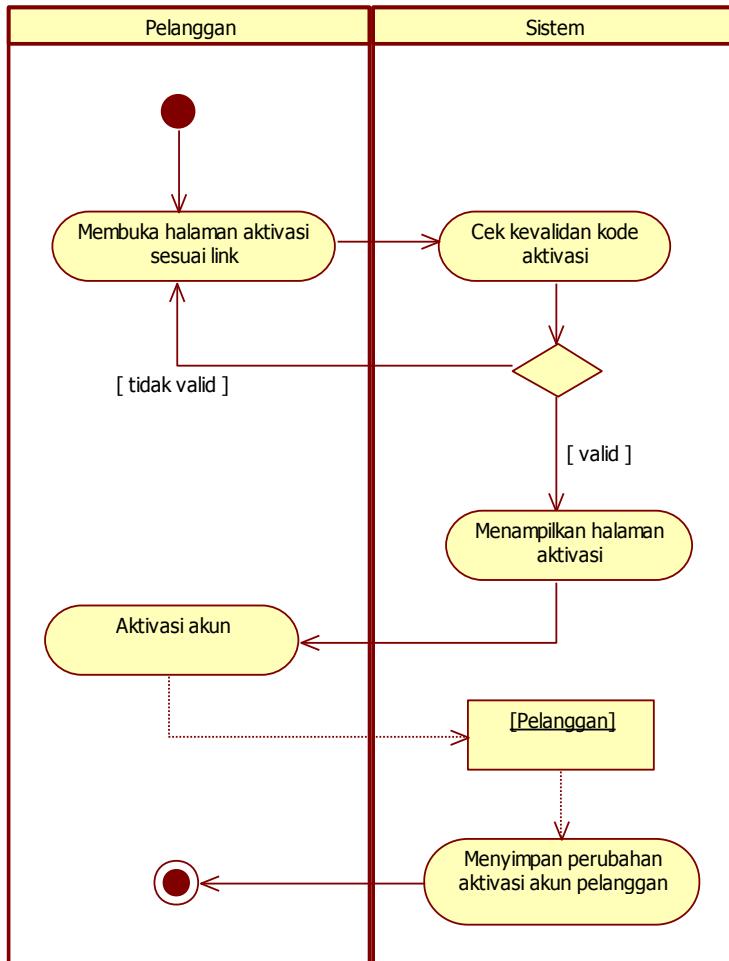
Gambar 4.41 Diagram Aktivitas untuk Kasus Melakukan Konfirmasi Pemesanan Bunga



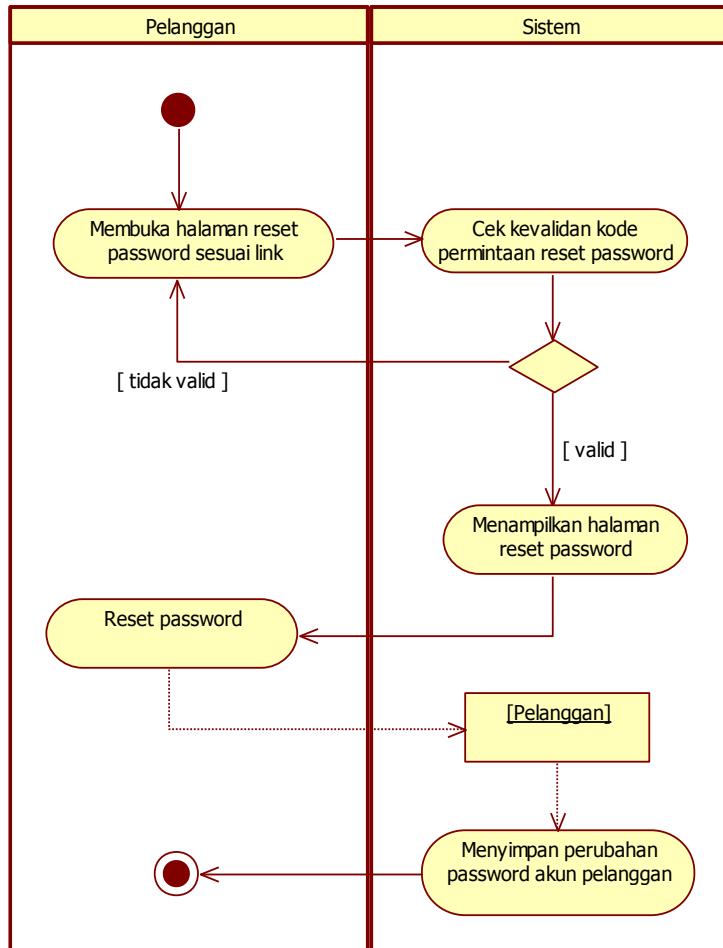
Gambar 4.42 Diagram Aktivitas untuk Kasus Mengirim Pemberitahuan kepada Pelanggan



Gambar 4.43 Diagram Aktivitas untuk Kasus Mengubah Profil dan Kontak Toko Meme Florist



Gambar 4.44 Diagram Aktivitas untuk Kasus Melakukan Aktivasi Akun Pelanggan



Gambar 4.45 Diagram Aktivitas untuk Kasus Mengubah Password Akun Pelanggan

- **Perancangan Data**

Basis data yang digunakan pada aplikasi *web* Meme Florist ini adalah Datastore dari Google App Engine dengan implementasi JDO (*Java Data Object*) versi 3.0 untuk penyimpanan data dengan tipe data pada umumnya dan *Blobstore* untuk menyimpan data dalam bentuk *file* yang ukurannya cukup besar seperti gambar. *Datastore* adalah salah satu jenis basis data yang bersifat *NoSQL* dan *schemaless*, sehingga dalam perancangan data ini tidak dibuatkan CDM (*Conceptual Data Modelling*) dan PDM (*Physical Data Modelling*), selain itu juga berbasiskan pada penggunaan *class* baik sebagai *parent* maupun *child* daripada menggunakan relasi seperti *foreign key*. Jika pada basis data pada dikenal sebagai tabel, maka di Datastore disebut dengan *kind*. Berdasarkan hasil analisa kebutuhan, dibutuhkan beberapa *kind* untuk menunjang proses bisnis yang ada. Berikut ini adalah beberapa *kind* yang dibutuhkan:

1. *Kind User*

Kind User adalah *kind* yang berisikan *entity* pelanggan Meme Florist. Atribut dari *User* ditunjukkan pada Tabel 4.1. Untuk alasan keamanan, *password* pelanggan harus dipisahkan ke dalam *kind* tersendiri.

Tabel 4.1 Atribut *Kind User*

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
fullname	String	Nama lengkap pelanggan
email	String	Alamat <i>email</i> pelanggan

Nama Variabel	Tipe Data	Keterangan
phone	String	Nomor <i>handphone</i> pelanggan
address	String	Alamat rumah pelanggan
activated	Boolean	Sebagai <i>flag</i> (penanda) bahwa akun sudah diaktivasi atau belum

2. *Kind Password*

Kind Password adalah *kind* yang berisikan *entity password* pelanggan Meme Florist. Atribut *Password* ditunjukkan pada Tabel 4.2

Tabel 4.2 Atribut *Kind Password*

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
password	String	Berisi <i>password</i> pelanggan yang dikodekan dengan algoritma PBKDF2 dengan Hmac SHA1
salt	String	Berisi <i>key</i> yang dibutuhkan untuk mendekripsi <i>password</i>
user	User	Menunjukkan pelanggan yang memiliki <i>password</i> bersangkutan

3. *Kind Category*

Kind Category adalah *kind* yang berisikan *entity* kategori bunga yang dijual oleh toko Meme Florist. Atribut *Category* ditunjukkan pada Tabel 4.3

Tabel 4.3 Atribut Kind Category

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
name	String	Kategori bunga

4. Kind Catalog

Kind Catalog adalah *kind* yang berisikan *entity* bunga yang dijual oleh toko Meme Florist. Atribut *Catalog* ditunjukkan pada Tabel 4.4.

Tabel 4.4 Atribut Kind Catalog

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
name	String	Nama bunga
width	String	Ukuran lebar bunga
height	String	Ukuran tinggi bunga
price	Long	Harga bunga
desc	String	Deskripsi dari bunga
imageId	String	Sebagai <i>Blob Key</i> dalam pengambilan data dari <i>Blobstore</i> yang berisi foto/gambar bunga
category	Category	Menunjukkan kategori bunga tersebut

5. Kind OrderItem

Kind OrderItem adalah *kind* yang berisikan *entity* sebuah *item* pemesanan, sehingga setiap kali pelanggan memesan dalam tiap transaksinya bisa memiliki lebih dari satu *item* pemesanan. Atribut *OrderItem* ditunjukkan pada Tabel 4.5.

Tabel 4.5 Atribut Kind OrderItem

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai Primary Key atau Unique ID
item	Catalog	Bunga yang dipesan
content	String	Teks yang akan dicantumkan pada bunga yang dipesan
amount	Long	Jumlah bunga yang dipesan

6. *Kind Order*

Kind Order adalah *kind* yang berisikan *entity* sekumpulan *OrderItem* dalam satu kali transaksi. Atribut *Order* ditunjukkan pada Tabel 4.6.

Tabel 4.6 Atribut Kind Order

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai Primary Key atau Unique ID
items	List of OrderItem	Beberapa <i>OrderItem</i> yang dipesan dalam satu transaksi
sender	User	Pelanggan yang melakukan transaksi
recipientName	String	Nama orang yang dituju
recipientAddress	String	Alamat orang yang dituju
recipientCity	String	Nama kota alamat yang dituju
date	Date	Tanggal pemesanan
totalPrice	Long	Total harga yang harus dibayar pelanggan
note	String	Catatan yang dilampirkan pelanggan

Nama Variabel	Tipe Data	Keterangan
status	Boolean	Status bahwa pemesanan telah dikirim atau belum

7. *Kind Shipment*

Kind Shipment adalah *kind* yang berisikan *entity* konfirmasi pengiriman pesanan bunga. Atribut *Shipment* ditunjukkan pada Tabel 4.7.

Tabel 4.7 Atribut *Kind Shipment*

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
order	Order	Pemesanan yang telah diproses dan dikirim
recipientName	String	Nama penerima barang di tempat tujuan
images	List of String	Berisi <i>Blob Key</i> yang digunakan untuk menampilkan lampiran foto bukti pengiriman dari <i>Blobstore</i>
note	String	Catatan pengiriman
date	Date	Tanggal konfirmasi pengiriman

8. *Kind Company*

Kind Company adalah *kind* yang berisikan *entity* profil toko Meme Florist. Atribut *Company* ditunjukkan pada Tabel 4.8.

Tabel 4.8 Atribut Kind Company

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
loginUsername	String	<i>Username</i> yang digunakan untuk <i>login administrator</i> di aplikasi web
password	String	<i>Password</i> yang digunakan untuk <i>login administrator</i> di aplikasi web
latitude	String	Koordinat lintang alamat toko untuk <i>Google Maps</i>
longitude	String	Koordinat bujur alamat toko untuk <i>Google Maps</i>
address	String	Alamat toko Meme Florist
about	Text	Tentang toko Meme Florist

9. *Kind CompanyContact*

Kind CompanyContact adalah *kind* yang berisikan *entity* kontak toko Meme Florist yakni alamat *email*, nomor telepon, pin *Blackberry*, rekening bank, dan lain sebagainya. Atribut *CompanyContact* ditunjukkan pada Tabel 4.9.

Tabel 4.9 Atribut Kind CompanyContact

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
type	String	Jenis kontak
value	String	Isi kontak

10. *Kind GCMDevices*

Kind GCMDevices adalah *kind* yang berisikan *entity* token dari sebuah *device* atau *handphone* untuk dapat menggunakan layanan *Google Cloud Messaging* (GCM). Hal ini agar aplikasi *web* dapat mengirimkan *push notification* kepada *device* pelanggan. Atribut *GCMDevices* ditunjukkan pada Tabel 4.10.

Tabel 4.10 Atribut *Kind GCMDevices*

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
deviceRegId	String	Token <i>device</i> pelanggan
user	User	Pelanggan yang memiliki <i>device</i>
acceptPersonalNotification	Boolean	<i>Flag</i> atau penanda bahwa pelanggan memperbolehkan notifikasi secara pribadi dari aplikasi <i>web</i> , biasanya berupa konfirmasi pengiriman barang
acceptBroadcastNotification	Boolean	<i>Flag</i> atau penanda bahwa pelanggan memperbolehkan notifikasi secara massal dari aplikasi <i>web</i> , biasanya berupa promosi

11. Kind Notification

Kind Notification adalah *kind* yang berisikan *entity* pesan pemberitahuan berupa *push notification* yang dikirimkan melalui aplikasi *web* dengan layanan *Google Cloud Messaging* (GCM) kepada *device* pelanggan. Atribut *Notification* ditunjukkan pada Tabel 4.11.

Tabel 4.11 Atribut Kind Notification

Nama Variabel	Tipe Data	Keterangan
id	Long	Sebagai <i>Primary Key</i> atau <i>Unique ID</i>
users	List of Long	Daftar pelanggan yang telah dikirim pesan ke <i>device</i> nya
message	String	Isi pesan
date	Date	Tanggal pengiriman pesan pemberitahuan

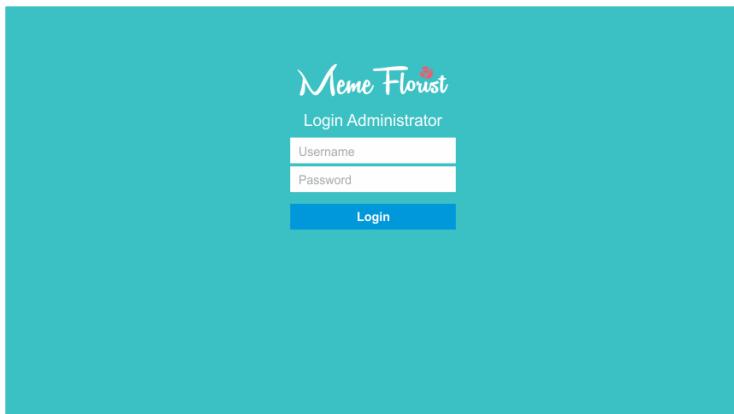
- **Perancangan Antar Muka**

- a. **Antarmuka Halaman Login Administrator Meme Florist**

Untuk halaman *Login* Administrator Meme Florist ini terdapat beberapa kontrol seperti rancangan pada Gambar 4.46. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk logo Meme Florist menggunakan kontrol *Image*
- Untuk isian *Username* menggunakan kontrol *Input type Text*
- Untuk isian *Password* menggunakan kontrol *Input type Password*

- Untuk tombol *Login* menggunakan kontrol *Button*



Gambar 4.46 Rancangan antarmuka halaman *login* administrator

b. Antarmuka Halaman Tampilan Daftar Katalog Bunga

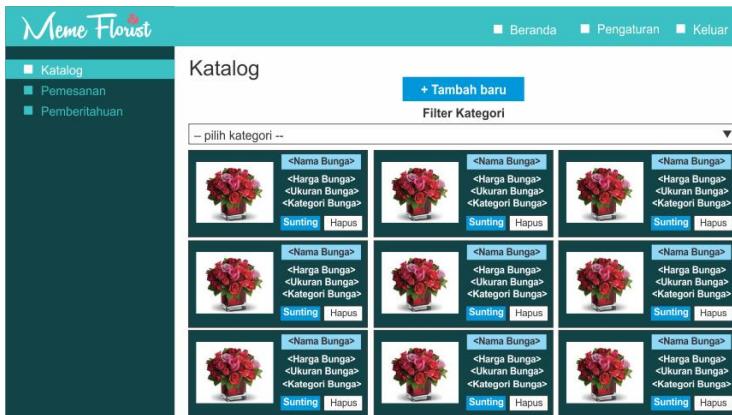
Untuk rancangan antarmuka halaman tampilan daftar katalog bunga ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.47. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk tombol Tambah baru menggunakan kontrol *Button*
- Untuk *filter* berdasarkan kategori menggunakan kontrol *Select*
- Untuk setiap katalog bunga memiliki beberapa kontrol seperti yang ditunjukkan oleh Gambar 4.48 dengan rincian sebagai berikut:

Rancang Bangun Aplikasi Meme Florist dengan Google App Engine Berbasis Perangkat Mobile Android untuk Pemesanan Bunga Online

Kantor PT. Tunas Sinergi Sejahtera (7Langit) Jakarta Jalan Moch.Yamin No.1 Jakarta Pusat - DKI Jakarta
23 Juni 2014 – 23 Juli 2014

1. Untuk gambar bunga menggunakan kontrol *Image*
2. Untuk atribut bunga menggunakan kontrol *Table*
3. Untuk tombol Sunting menggunakan kontrol *Button*
4. Untuk tombol Hapus menggunakan kontrol *Button*



The screenshot shows the 'Katalog' (Catalog) screen of the Meme Florist app. At the top, there's a navigation bar with icons for Home, Settings, and Logout. Below the bar, a title 'Katalog' is displayed next to a '+ Tambah baru' (Add new) button. A 'Filter Kategori' (Category Filter) dropdown is also present. The main area contains a grid of nine flower arrangements, each with a 'Nama Bunga' (Flower Name) label above it and 'Sunting' (Edit) and 'Hapus' (Delete) buttons below. Each arrangement is accompanied by its name and a small image.

Gambar 4.47 Rancangan antarmuka halaman daftar katalog bunga

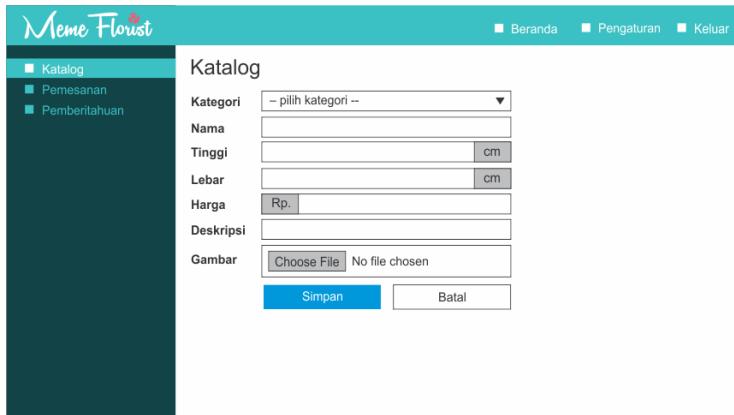


Gambar 4.48 Rancangan antarmuka *item* katalog bunga

c. Antarmuka Halaman Menambahkan Data Katalog Bunga

Untuk rancangan antarmuka halaman menambahkan data katalog bunga terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.49. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk isian pilih kategori menggunakan kontrol *Select*
- Untuk isian Nama menggunakan kontrol *Input type Text*
- Untuk isian Tinggi menggunakan kontrol *Input type Text*
- Untuk isian Lebar menggunakan kontrol *Input type Text*
- Untuk isian Harga menggunakan kontrol *Input type Number*
- Untuk isian Deskripsi menggunakan kontrol *Input type Text*
- Untuk isian Gambar menggunakan kontrol *Input type File*
- Untuk tombol Simpan menggunakan kontrol *Button*
- Untuk tombol Batal menggunakan kontrol *Button*



The screenshot shows a mobile application interface for 'Meme Florist'. At the top, there's a navigation bar with three items: 'Beranda' (Home), 'Pengaturan' (Settings), and 'Keluar' (Logout). Below this is a header with the app's name 'Meme Florist' and a small flower icon. On the left, a sidebar menu has three items: 'Katalog' (selected), 'Pemesanan', and 'Pemberitahuan'. The main content area is titled 'Katalog' and contains several input fields: 'Kategori' (dropdown menu with placeholder '- pilih kategori -'), 'Nama' (text input), 'Tinggi' (text input with unit 'cm' next to it), 'Lebar' (text input with unit 'cm' next to it), 'Harga' (text input with placeholder 'Rp.'), 'Deskripsi' (text input), and 'Gambar' (file upload button with placeholder 'Choose File' and message 'No file chosen'). At the bottom are two buttons: a blue 'Simpan' (Save) button and a white 'Batal' (Cancel) button.

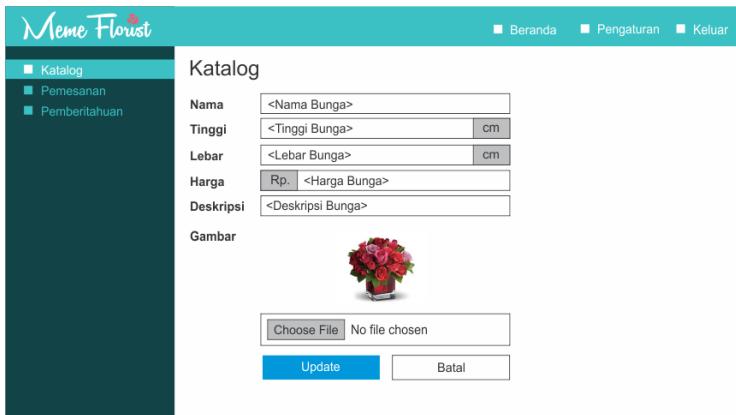
Gambar 4.49 Rancangan antarmuka menambah data katalog bunga

d. Antarmuka Halaman Mengubah Data Katalog Bunga

Untuk rancangan antarmuka halaman mengubah data katalog bunga terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.50. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk isian Nama menggunakan kontrol *Input type Text*
- Untuk isian Tinggi menggunakan kontrol *Input type Text*
- Untuk isian Lebar menggunakan kontrol *Input type Text*
- Untuk isian Harga menggunakan kontrol *Input type Number*
- Untuk isian Deskripsi menggunakan kontrol *Input type Text*

- Untuk tampilan Gambar menggunakan kontrol *Image*
- Untuk isian Gambar menggunakan kontrol *Input type File*
- Untuk tombol *Update* menggunakan kontrol *Button*
- Untuk tombol Batal menggunakan kontrol *Button*



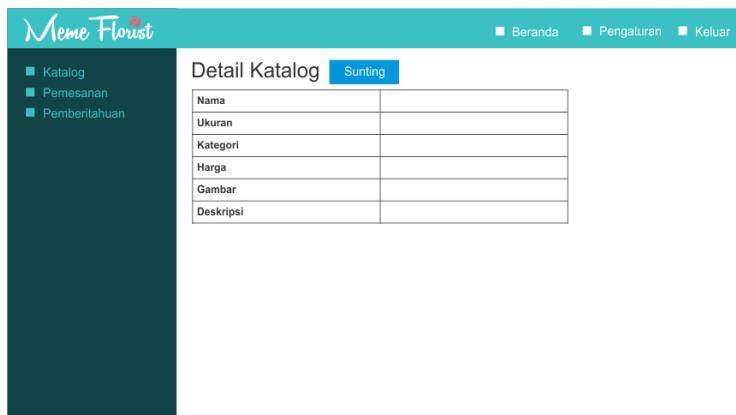
Gambar 4.50 Rancangan antarmuka mengubah data katalog bunga

e. Antarmuka Halaman Tampilan Detail Katalog Bunga

Untuk rancangan antarmuka halaman tampilan detail katalog bunga ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.51. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk tombol Sunting menggunakan kontrol *Button*

- Untuk tampilan detail atribut katalog menggunakan kontrol *Table* dengan isi semuanya dalam bentuk *Text* kecuali untuk atribut Gambar menggunakan kontrol *Image*

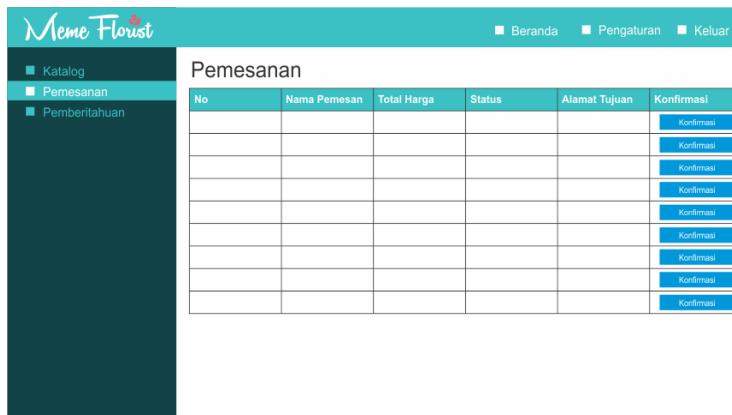


Gambar 4.51 Rancangan antarmuka halaman detail katalog

f. Antarmuka Halaman Tampilan Daftar Pemesanan Pelanggan

Untuk rancangan antarmuka halaman tampilan daftar pemesanan pelanggan ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.52. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk tampilan daftar pemesanan pelanggan menggunakan kontrol *Table* dengan semua isinya dalam bentuk *Text* kecuali atribut Konfirmasi yang menggunakan kontrol *Button*.



The screenshot shows a mobile application interface for 'Meme Florist'. The top navigation bar includes links for 'Beranda', 'Pengaturan', and 'Keluar'. On the left, a sidebar menu lists 'Katalog', 'Pemesanan' (which is highlighted in blue), and 'Pemberitahuan'. The main content area is titled 'Pemesanan' and displays a table with 10 rows of order data. The columns are labeled 'No', 'Nama Pemesan', 'Total Harga', 'Status', 'Alamat Tujuan', and 'Konfirmasi'. Each row contains a 'Konfirmasi' button at the bottom right.

No	Nama Pemesan	Total Harga	Status	Alamat Tujuan	Konfirmasi
1					Konfirmasi
2					Konfirmasi
3					Konfirmasi
4					Konfirmasi
5					Konfirmasi
6					Konfirmasi
7					Konfirmasi
8					Konfirmasi
9					Konfirmasi
10					Konfirmasi

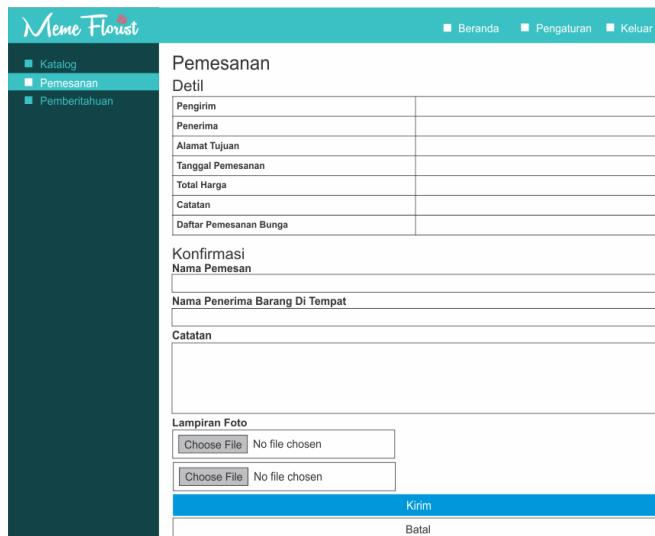
Gambar 4.52 Rancangan antarmuka menampilkan daftar pemesanan pelanggan

g. Antarmuka Halaman Konfirmasi Pemesanan Pelanggan

Untuk rancangan antarmuka halaman konfirmasi pemesanan pelanggan ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.53. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk tampilan detail pemesanan pelanggan menggunakan kontrol *Table* dengan isi semuanya dalam bentuk *Text*.
- Untuk isian Nama Pemesan menggunakan kontrol *Input type Text*
- Untuk isian Nama Penerima Barang Di Tempat menggunakan kontrol *Input type Text*
- Untuk isian Catatan menggunakan kontrol *Textarea*

- Untuk isian Lampiran menggunakan kontrol *Input type File*
- Untuk tombol Kirim menggunakan kontrol *Button*
- Untuk tombol Batal menggunakan kontrol *Button*



The screenshot shows the 'Meme Florist' mobile application interface. At the top, there is a navigation bar with three items: 'Beranda' (Home), 'Pengaturan' (Settings), and 'Keluar' (Logout). On the left side, there is a vertical sidebar with three menu items: 'Katalog' (Catalog), 'Pemesanan' (Order), and 'Pemberitahuan'. The main content area is titled 'Pemesanan Detil'. It contains several input fields for order details: 'Pengirim', 'Penerima', 'Alamat Tujuan', 'Tanggal Pemesanan', 'Total Harga', 'Catatan', and 'Daftar Pemesanan Bunga'. Below these fields is a section titled 'Konfirmasi' with two input fields: 'Nama Pemesan' and 'Nama Penerima Barang Di Tempat'. There is also a large 'Catatan' input field. Further down is a section titled 'Lampiran Foto' with two 'Choose File' buttons. At the bottom of the screen, there are two buttons: 'Kirim' (Send) in blue and 'Batal' (Cancel) in white.

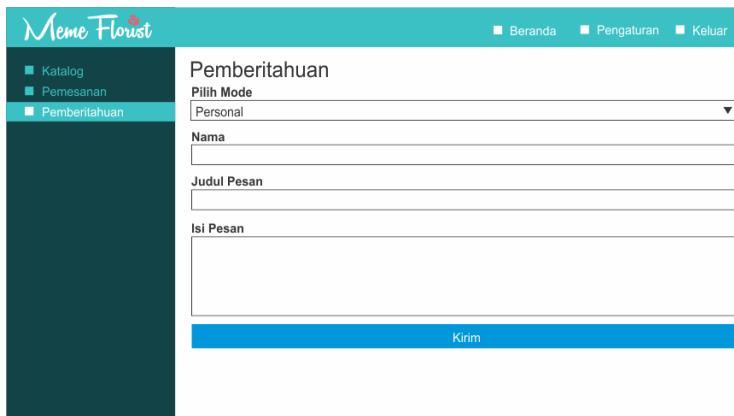
Gambar 4.53 Rancangan antarmuka konfirmasi pemesanan pelanggan

h. Antarmuka Halaman Pemberitahuan Kepada Pelanggan

Untuk rancangan antarmuka halaman pemberitahuan kepada pelanggan ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar

4.54. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk isian Pilih Mode menggunakan kontrol *Select*
- Untuk isian Nama menggunakan kontrol *Input type Text*
- Untuk isian Judul Pesan menggunakan kontrol *Input type Text*
- Untuk isian Isi Pesan menggunakan kontrol *Textarea*
- Untuk tombol Kirim menggunakan kontrol *Button*



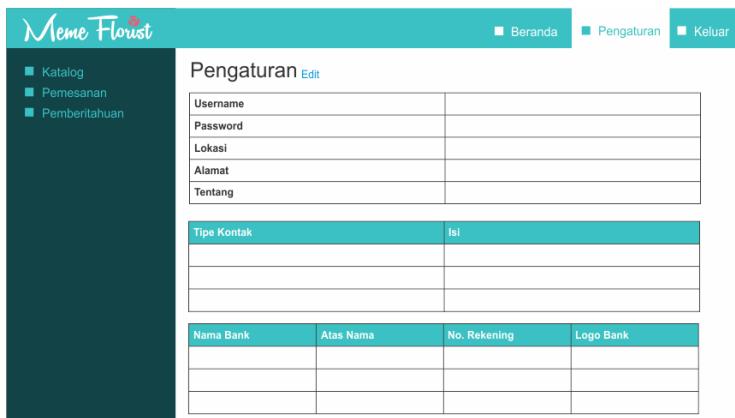
Gambar 4.54 Rancangan antarmuka mengirim pemberitahuan kepada pelanggan

i. **Antarmuka Halaman Tampilan Pengaturan Profil dan Kontak Toko**

Untuk rancangan antarmuka halaman tampilan pengaturan profil dan kontak toko Meme

Florist ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.55. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk tampilan atribut profil toko menggunakan kontrol *Table* dengan isi semuanya dalam bentuk *Text*
- Untuk tampilan atribut kontak (selain rekening bank) menggunakan kontrol *Table* dengan isi semuanya dalam bentuk *Text*
- Untuk tampilan atribut kontak (khusus rekening bank) menggunakan kontrol *Table* dengan isi semuanya dalam bentuk *Text* kecuali atribut Logo Bank menggunakan kontrol *Image*



The screenshot shows the 'Pengaturan' (Settings) screen of the Meme Florist app. At the top, there's a navigation bar with 'Beranda', 'Pengaturan' (highlighted in blue), and 'Keluar'. On the left, a sidebar menu lists 'Katalog', 'Pemesanan', and 'Pemberitahuan'. The main content area contains three tables: 1) 'Pengaturan' with fields for Username, Password, Lokasi, Alamat, and Tentang. 2) 'Tipe Kontak' with columns for Tipe Kontak and Isi. 3) 'Nama Bank' with columns for Nama Bank, Atas Nama, No. Rekening, and Logo Bank.

Tipe Kontak	Isi

Nama Bank	Atas Nama	No. Rekening	Logo Bank

Gambar 4.55 Rancangan antarmuka menampilkan pengaturan profil dan kontak Meme Florist

j. Antarmuka Halaman Pengubahan Profil dan Kontak Toko

Untuk rancangan antarmuka halaman pengubahan profil dan kontak toko Meme Florist ini terdiri dari beberapa kontrol yang ditunjukkan oleh Gambar 4.56. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk isian *Username* menggunakan kontrol *Input type Text*
- Untuk isian *Password* menggunakan kontrol *Input type Password*
- Untuk tampilan peta menggunakan kontrol *embed* dari *Google Maps*
- Untuk isian Cari Tempat menggunakan kontrol *Input type Text*
- Untuk tombol Cari menggunakan kontrol *Button*
- Untuk isian Latitude menggunakan kontrol *Input type Text*
- Untuk isian Longitude menggunakan kontrol *Input type Text*
- Untuk isian Kontak terdiri dari beberapa kontrol dan bersifat dinamis, artinya dapat ditambah dan dikurangi dari halaman. Rincian kontrol yang digunakan antara lain:
 1. Untuk jenis kontak khusus Account Number terdiri dari beberapa kontrol, antara lain sebagai berikut:
 - Untuk pilihan jenis kontak menggunakan kontrol *Select*

- Untuk isian kontak menggunakan kontrol *Input type Text*
 - Untuk tombol dengan simbol tambah (+) menggunakan kontrol *Button*
 - Untuk tombol dengan simbol kurang (-) menggunakan kontrol *Button*
2. Untuk jenis kontak selain Account Number terdiri dari beberapa kontrol, antara lain sebagai berikut:
- Untuk pilihan jenis kontak menggunakan kontrol *Select*
 - Untuk isian Nama Bank menggunakan kontrol *Input type Text*
 - Untuk isian Atas Nama menggunakan kontrol *Input type Text*
 - Untuk isian Nomor Rekening menggunakan kontrol *Input type Text*
 - Untuk isian Logo Bank menggunakan kontrol *Input type File*
 - Untuk tombol dengan simbol tambah (+) menggunakan kontrol *Button*
 - Untuk tombol dengan simbol kurang (-) menggunakan kontrol *Button*

k. Antarmuka Halaman Aktivasi Akun Pelanggan

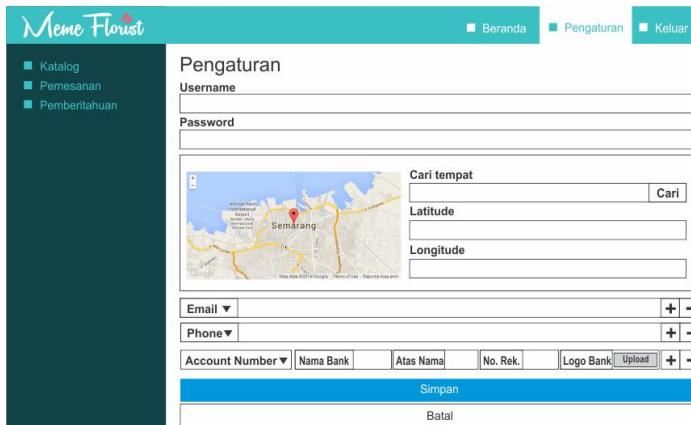
Untuk rancangan antarmuka halaman aktivasi akun pelanggan hanya menampilkan notifikasi hasil aktivasi karena kunci dari aktivasi akun pelanggan ini tergantung pada *link* aktivasinya, sehingga tidak terdapat banyak

kontrol dibutuhkan. Kontrol yang ada antara lain sebagai berikut:

- Untuk logo Meme Florist menggunakan kontrol *Image*
- Untuk tombol Kunjungi Kami menggunakan kontrol *Button*

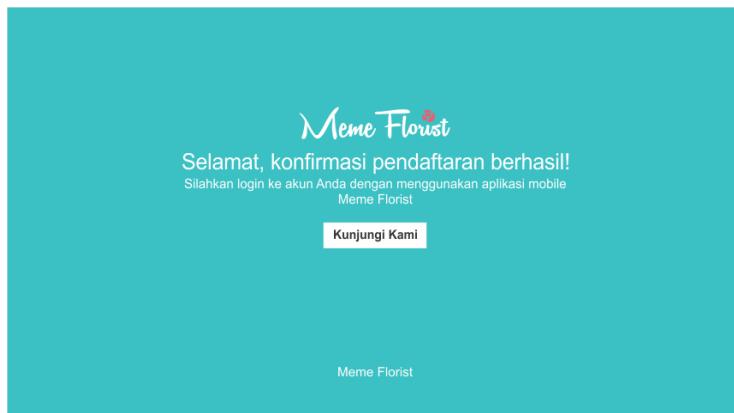
Terdapat 2 macam hasil dari aktivasi akun pengguna ini, antara lain sebagai berikut:

- Jika kode aktivasi di dalam *link* dinyatakan valid, maka akan menampilkan halaman notifikasi seperti yang ditunjukkan oleh Gambar 4.57.
- Jika kode aktivasi di dalam *link* dinyatakan tidak valid, maka akan menampilkan halaman notifikasi seperti yang ditunjukkan oleh Gambar 4.58.

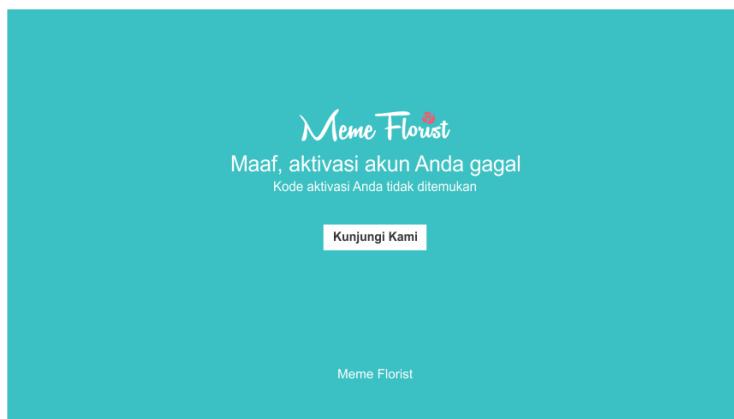


The screenshot shows the 'Pengaturan' (Settings) screen of the Meme Florist app. At the top, there are three navigation buttons: 'Beranda' (Home), 'Pengaturan' (Settings), and 'Keluar' (Logout). On the left, a sidebar menu lists 'Katalog', 'Pemesanan', and 'Pemberitahuan'. The main area contains fields for 'Username' and 'Password'. Below these are sections for location settings ('Cari tempat', 'Latitude', 'Longitude'), and contact information ('Email', 'Phone', 'Account Number', 'Nama Bank', 'Atas Nama', 'No. Rek.', 'Logo Bank', 'Upload'). At the bottom are 'Simpan' (Save) and 'Batal' (Cancel) buttons.

Gambar 4.56 Rancangan antarmuka pengubahan profil dan kontak Meme Florist



Gambar 4.57 Rancangan antarmuka halaman notifikasi sukses aktivasi akun pelanggan



Gambar 4.58 Rancangan antarmuka halaman notifikasi gagal aktivasi akun pelanggan

I. Antarmuka Halaman *Reset Password* Akun Pelanggan

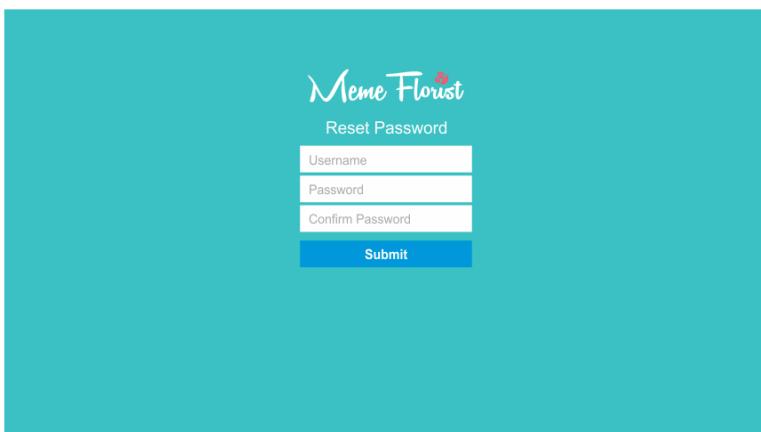
Untuk rancangan antarmuka halaman *reset password* akun pelanggan ini tergantung pada *link* aktivasinya. Terdapat 2 macam hasil dari aktivasi akun pengguna ini, antara lain sebagai berikut:

- Jika kode permintaan *reset password* di dalam *link* dinyatakan valid, maka akan menampilkan seperti yang ditunjukkan oleh Gambar 4.59. Kontrol yang digunakan antara lain sebagai berikut:
 - Untuk logo Meme Florist menggunakan kontrol *Image*
 - Untuk isian *Username* menggunakan kontrol *Input type Text*
 - Untuk isian *Password* menggunakan kontrol *Input type Password*
 - Untuk isian *Confirm Password* menggunakan kontrol *Input type Password*
 - Untuk tombol *Submit* menggunakan kontrol *Button*

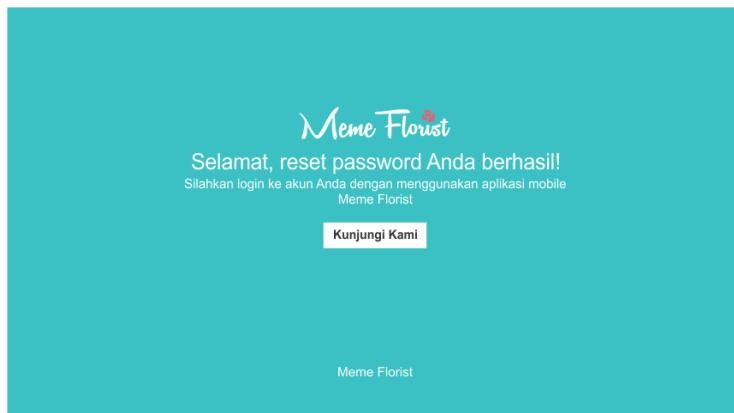
Kemudian jika *reset password* sukses, akan menampilkan halaman notifikasi seperti yang ditunjukkan oleh Gambar 4.60 yang di dalamnya terdapat beberapa kontrol. Kontrol yang digunakan antara lain sebagai berikut:

- Untuk logo Meme Florist menggunakan kontrol *Image*

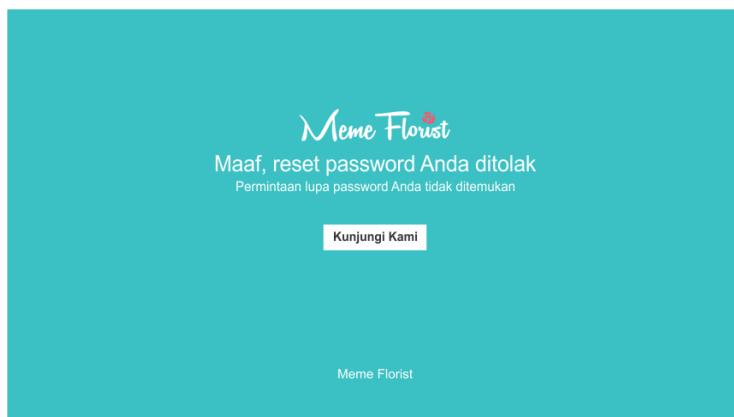
- Untuk tombol Kunjungi Kami menggunakan kontrol *Button*
- Untuk teks Meme Florist merupakan *link* yang menggunakan kontrol *Hyperlink*
- Jika kode permintaan *reset password* di dalam *link* dinyatakan tidak valid, maka akan menampilkan halaman notifikasi seperti yang ditunjukkan oleh Gambar 4.61. Kontrol yang digunakan antara lain sebagai berikut:
 - Untuk logo Meme Florist menggunakan kontrol *Image*
 - Untuk tombol Kunjungi Kami menggunakan kontrol *Button*
 - Untuk teks Meme Florist merupakan *link* yang menggunakan kontrol *Hyperlink*



Gambar 4.59 Rancangan antarmuka halaman *reset password* akun pelanggan



Gambar 4.60 Rancangan antarmuka notifikasi sukses *reset password* akun pelanggan



Gambar 4.61 Rancangan antarmuka notifikasi gagal *reset password* akun pelanggan

BAB V

IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi berupa kode program yang digunakan untuk membangun aplikasi Meme Florist dengan Google App Engine berbasis perangkat *Mobile Android* untuk pemesanan bunga *online*.

5.1 Implementasi Aplikasi Meme Florist

Di bagian ini akan dijelaskan mengenai implementasi berupa kode program yang digunakan untuk membangun aplikasi *mobile* Meme Florist. Kode program telah terbagi menjadi beberapa bagian baik untuk tampilan di pengguna (.xml) dan administrator (.html) maupun kode di belakang (*code behind*) yaitu berbahasa Java (.java). Dalam tiap-tiap bagian kode akan diberikan keterangan mengenai maksud dari kode tersebut. Secara detail mengenai implementasi aplikasi Meme Florist dijabarkan sebagai berikut:

5.1.1. Melihat kategori bunga dan detil bunga

- **Tampilan fragment_category_katalog.xml**

Pada tampilan ini terdapat kontrol *GridView* untuk menampilkan kategori bunga secara keseluruhan. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.1.

```
11   <GridView  
12     android:id="@+id/gridView1"  
13     android:layout_width="match_parent"  
14     android:layout_height="match_parent"  
15     android:horizontalSpacing="5dp"  
16     android:numColumns="2"  
17     android:stretchMode="columnWidth"  
18     android:layout_below="@+id/textView1"  
19     android:verticalSpacing="5dp"  
20     android:listSelector="@android:color/transparent" >  
21   </GridView>
```

Gambar 5.1 Kode program fragment_category_katalog.xml

- **Kelas KatalogFragment.java**

Kelas ini merupakan *code behind* dari tampilan KatalogFragment.java. Fungsi yang terkait dengan pengembangan yang dimaksud ada dalam beberapa potongan bagian kode ini.

- a. **Fungsi doInBackground(Void... unused)**

Fungsi ini digunakan untuk mengeksekusi kode dalam melakukan proses *query* seluruh kategori bunga di *background* ketika halaman sedang dimuat. Inti dari fungsi ini ditunjukkan oleh Gambar 5.2 baris 94.

- b. **Fungsi onPostExecute(CategoryCollection categories)**

Fungsi ini digunakan untuk mengeksekusi kode setelah proses yang berada di *background* selesai. Hasil query yang didapatkan, dilakukan proses untuk menampilkannya dalam bentuk *GridView*. Inti dari fungsi ini ditunjukkan oleh Gambar 5.3 baris 111.

```
84@  protected CategoryCollection doInBackground(Void... unused) {  
85      // android.os.Debug.waitForDebugger();  
86      CategoryCollection categories = null;  
87      try {  
88          Log.d("Try", "Masuk");  
89          FloristAPI.Builder builder = new FloristAPI.Builder(  
90              AndroidHttp.newCompatibleTransport(),  
91              new AndroidJsonFactory(), null);  
92          builder.setRootUrl(Constant.URL);  
93          FloristAPI service = builder.build();  
94          categories = service.getCategories().execute();  
95          Log.d("Size", String.valueOf(categories));  
96      } catch (Exception e) {  
97          Log.d("Could not retrieve Catalog", e.getMessage(), e);  
98      }  
99      return categories;  
100 }
```

Gambar 5.2 Kode fungsi doInBackground asynctask KatalogFragment

```
102    protected void onPostExecute(CategoryCollection categories) {  
103        progressBar.setProgressBar(View.GONE);  
104        Log.d("TES", "ONPOST");  
105        listCategory = categories.getItems();  
106        Log.d("DEBUG", String.valueOf(listCategory.size()));  
107        List<Long> idCategory = new ArrayList<Long>();  
108        for (Category c : listCategory) {  
109            idCategory.add(c.getId());  
110        }  
111        gridView.setOnItemClickListener(new OnItemClickListener() {  
112            @Override  
113            public void onItemClick(AdapterView<?> parent, View v,  
114                int position, long id) {  
115                Intent i = new Intent(getActivity(),  
116                    PagerViewBungaActivity.class);  
117                i.putExtra("id", position);  
118                i.putExtra("idCategory", listCategory.get(position).getId());  
119                startActivity(i);  
120            }  
121        });  
122        mTask = null;  
123    }  
124}
```

Gambar 5.3 Kode fungsi onPostExecute asynctask KatalogFragment

- **Tampilan activity_page_bunga.xml**

Pada tampilan ini terdapat kontrol *ViewPager* untuk menampilkan bunga secara keseluruhan pada masing-masing kategori. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.4.

```
18    <android.support.v4.view.ViewPager  
19        android:id="@+id/pager"  
20        android:layout_below="@+id/indicator"  
21        android:layout_width="fill_parent"  
22        android:layout_height="fill_parent" />
```

Gambar 5.4 Kode program activity_page_bunga.xml

- **Kelas PagerViewBungaActivity.java**

6.1 Fungsi doInBackground (Long... idCategory)

Fungsi ini digunakan untuk mengeksekusi kode dalam melakukan proses *query* kategori

bunga yang telah dipilih di *background* ketika halaman sedang dimuat. Inti dari fungsi ini ditunjukkan oleh Gambar 5.5 baris 430.

6.2 Fungsi `onPostExecute(CatalogCollection catalogs)`

Fungsi ini digunakan untuk mengeksekusi kode setelah proses yang berada di *background* selesai. Hasil query yang didapatkan, dilakukan proses untuk menampilkannya dalam bentuk *PagerView*. Inti dari fungsi ini ditunjukkan oleh Gambar 5.6 baris 465.

5.1.2. Melihat informasi tentang profil dan kontak Meme Florist

- Kelas `TentangFloristFragment.java`**

Pada kelas ini terdapat fungsi `Company` yang digunakan untuk mengeksekusi kode dalam melakukan proses *query* informasi tentang Meme Florist di *background* ketika halaman sedang dimuat. Inti dari fungsi ini ditunjukkan oleh Gambar 5.7 baris 107.

- Tampilan `activity_hubungi_kami.xml`**

Pada tampilan ini terdapat kontrol *ListView* untuk menampilkan kontak yang dapat digunakan untuk menghubungi Meme Florist. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.8.

- Kelas `HubungiKamiFragment.java`**

Pada kelas ini terdapat fungsi `List<CompanyContact> doInBackground(Void... unused)` yang digunakan untuk mengeksekusi kode dalam melakukan proses *query* informasi tentang kontak milik Meme Florist di *background* ketika halaman sedang dimuat. Inti dari fungsi ini ditunjukkan oleh Gambar 5.9 baris 187.

Rancang Bangun Aplikasi Meme Florist dengan Google App Engine Berbasis Perangkat Mobile Android untuk Pemesanan Bunga Online

Kantor PT. Tunas Sinergi Sejahtera (7Langit) Jakarta Jalan Moch.Yamin No.1 Jakarta Pusat - DKI Jakarta

23 Juni 2014 – 23 Juli 2014

```

420@    protected CatalogCollection doInBackground(Long... idCategory) {
421        // android.os.Debug.waitForDebugger();
422        catalogs = null;
423        try {
424            Log.d("Try", "Masuk");
425            FloristAPI.Builder builder = new FloristAPI.Builder(
426                AndroidHttp.newCompatibleTransport(),
427                new AndroidJsonFactory(), null);
428            builder.setRootUrl("https://skilled-curve-631.appspot.com/_ah/api");
429            FloristAPI service = builder.build();
430            catalogs = service.getCatalogsByCategory(idCategory[0])
431                .execute();
432            // List<Order> o = service.historyOrders(123L).execute();
433            Log.d("Size", String.valueOf(catalogs));
434        } catch (Exception e) {
435            Log.d("Could not retrieve Catalog", e.getMessage(), e);
436        }
437        return catalogs;
438    }

```

**Gambar 5.5 Kode fungsi doInBackground asynctask
PagerBungaActivity**

```

440@    protected void onPostExecute(CatalogCollection catalogs) {
441        progressBar.setVisibility(View.GONE);
442        listCatalog = catalogs.getItems();
443        hargai = new ArrayList<Long>();
444        ukurani = new ArrayList<String>();
445        img1 = new ArrayList<String>();
446        kodeBungai = new ArrayList<String>();
447        descripts1 = new ArrayList<String>();
448        idKatalog = new ArrayList<Long>();
449        for (Catalog c : listCatalog) {
450            hargai.add(c.getPrice());
451            hargai.add(c.getPrice());
452            ukurani.add(c.getWidth() + " x " + c.getHeight());
453            img1.add(Constant.URL_PIC + "serve?blob-key=" + c.getImageId());
454            kodeBungai.add(c.getName());
455            descripts1.add(c.getDesc());
456            idKatalog.add(c.getId());
457        }
458        try {
459            Bundle b = new Bundle();
460            Intent i = getIntent();
461            int position = i.getExtras().getInt("id");
462            pager = (ViewPager) findViewById(R.id.pager);
463            int pagerPos = 0;
464
465            pager.setAdapter(new ImagePagerAdapter(img1, hargai, ukurani,
466                kodeBungai, descripts1));
467            pager.setCurrentItem(pagerPos);
468
469            mIndicator = (CirclePageIndicator) findViewById(R.id.indicator);
470            mIndicator.setViewPager(pager);
471        } catch (NullPointerException e) {
472            Log.e("TAG", "TES");
473        }
474        mTask = null;
475    }

```

**Gambar 5.6 Kode fungsi onPostExecute asynctask
PagerBungaActivity**

```

97    protected Company doInBackground(Void... unused) {
98        // android.os.Debug.waitForDebugger();
99        Company companyReturn = null;
100       try {
101           Log.d("Try", "Masuk");
102           FloristAPI.Builder builder = new FloristAPI.Builder(
103               AndroidHttp.newCompatibleTransport(),
104               new AndroidJsonFactory(), null);
105           builder.setRootUrl(Constant.URL);
106           FloristAPI service = builder.build();
107           companyReturn = service.getCompanyProfile().execute();
108           // ifLog.d("Size", String.valueOf(userReturn));
109       } catch (Exception e) {
110           Log.d("Could not retrieve Catalog", e.getMessage(), e);
111       }
112       return companyReturn;
113   }

```

Gambar 5.7 Kode fungsi doInBackground asynctask kelas Company

```

30    <ListView
31        android:id="@+id/listView1"
32        android:layout_width="match_parent"
33        android:layout_height="wrap_content"
34        android:layout_below="@+id/textViewJudul"
35        android:layout_above="@+id/button1"
36        android:background="@android:color/transparent" >
37

```

Gambar 5.8 Kode program activity_hubungi_kami.xml

```

177    protected List<CompanyContact> doInBackground(Void... unused) {
178        // android.os.Debug.waitForDebugger();
179        List<CompanyContact> companyReturn = null;
180        try {
181            Log.d("Try", "Masuk");
182            FloristAPI.Builder builder = new FloristAPI.Builder(
183                AndroidHttp.newCompatibleTransport(),
184                new AndroidJsonFactory(), null);
185            builder.setRootUrl(Constant.URL);
186            FloristAPI service = builder.build();
187            companyReturn = service.getCompanyContacts().execute()
188                .getItems();
189            // ifLog.d("Size", String.valueOf(userReturn));
190        } catch (Exception e) {
191            Log.d("Could not retrieve Catalog", e.getMessage(), e);
192        }
193        return companyReturn;
194    }

```

Gambar 5.9 Kode fungsi doInBackground asynctask HubungiKamiFragment

- **Tampilan cara_pembayaran.xml**

Pada tampilan ini terdapat kontrol *ListView* untuk menampilkan nomor rekening yang dimiliki Meme Florist untuk keperluan pembayaran bunga yang telah dipesan oleh pengguna. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.10.

```
36     <ListView
37         android:id="@+id/listViewCaraBayar"
38         android:layout_width="match_parent"
39         android:layout_height="match_parent"
40         android:layout_marginTop="10dp"
41         android:layout_below="@+id/caraBayar" >
42     </ListView>
```

Gambar 5.10 Kode program cara_pembayaran.xml

- **Kelas CaraPembayaran.java**

Pada kelas ini terdapat fungsi *onCreate* yang digunakan untuk mengolah hasil eksekusi kode *query* pada fungsi *List<CompanyContact> doInBackground(Void... unused)* seperti pada Gambar 5.9. Hasil *query* yang didapatkan tersebut, dilakukan proses untuk menampilkannya dalam bentuk *ListView*. Inti dari fungsi ini ditunjukkan oleh Gambar 5.11 baris 67.

- **Tampilan activity_location_maps.xml**

Pada tampilan ini terdapat kontrol *fragment* untuk menampilkan peta lokasi Meme Florist yang berasal dari Google Maps. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.12.

- **Kelas LoadMaps.java**

Pada kelas ini terdapat fungsi *initializeMap()* yang digunakan untuk menginisialisasi peta lokasi Meme Florist berdasarkan *latitude* dan *longitude* untuk kemudian dipetakan ke dalam *fragment* yang telah disiapkan seperti yang ditunjukkan pada Gambar 5.12. Fungsi ini ditunjukkan oleh Gambar 5.13.

Rancang Bangun Aplikasi Meme Florist dengan Google Cloud Engine Berbasis Perangkat Mobile Android untuk Pemesanan Bunga Online

Kantor PT. Tunas Sinergi Sejahtera (7Langit) Jakarta Jalan Moch.Yamin No.1 Jakarta Pusat - DKI Jakarta

23 Juni 2014 – 23 Juli 2014

```

40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.activity_cara_pembayaran);
43
44         imageLoader = ImageLoader.getInstance();
45         imageLoader.init(ImageLoaderConfiguration.createDefault(this));
46
47         options = new DisplayImageOptions.Builder()
48             .showImageForEmptyUri(R.drawable.ic_empty)
49             .showImageOnFail(R.drawable.ic_error)
50             .resetViewBeforeLoading(true).cacheOnDisc(true)
51             .imageScaleType(ImageScaleType.EXACTLY)
52             .bitmapConfig(Bitmap.Config.RGB_565).considerExifParams(true)
53             .displayer(new FadeInBitmapDisplayer(300)).build();
54
55         listViewBank = (ListView) findViewById(R.id.ListViewCaraBayar);
56
57         companyContact = Constant.companyContact;
58         Log.d("companyContact", companyContact.toString());
59         List<String> bank = new ArrayList<String>();
60         for(CompanyContact c: companyContact){
61             if(c.getType().equals("accountNumber")){
62                 bank.add(c.getValue());
63             }
64             Log.d("BANK", bank.toString());
65         }
66         CompanyContactAdapter adapter = new CompanyContactAdapter(this, bank);
67         listViewBank.setAdapter(adapter);
68     }

```

Gambar 5.11 Kode fungsi onCreate

```

7     <fragment
8         android:id="@+id/map"
9         android:name="com.google.android.gms.maps.MapFragment"
10        android:layout_width="match_parent"
11        android:layout_height="match_parent"/>

```

Gambar 5.12 Kode program activity_location_maps.xml

```

47     private void initializeMap() {
48         if (googleMap == null) {
49             googleMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();
50
51             // check if map is created successfully or not
52             if (googleMap == null) {
53                 Toast.makeText(getApplicationContext(),
54                     "Sorry! unable to create maps", Toast.LENGTH_SHORT).show();
55             }
56         } else{
57             // latitude and longitude
58             Log.d("latlong", Constant.companyProfile.getLatitude().toString()+" "+Constant.companyProfile.getLatitude().toString());
59             double latitude = Double.parseDouble(Constant.companyProfile.getLatitude().toString());
60             double longitude = Double.parseDouble(Constant.companyProfile.getLongitude().toString());
61
62             // create marker
63             MarkerOptions marker = new MarkerOptions().position(new LatLng(latitude, longitude)).title("Meme Florist");
64             googleMap.addMarker(marker);
65         }
66     }

```

Gambar 5.13 Kode fungsi initializeMap

5.1.3. Mengelola keranjang belanja

- **Tampilan Kelas PagerViewActivity.java**

Pada kelas ini terdapat fungsi onClick() yang berada pada kontrol *AlertDialog* yang digunakan untuk menambah keranjang belanja pengguna. Kelas keranjang digunakan untuk kelas penyimpan keranjang belanja sementara di dalam *device*. Inti fungsi ini ditunjukkan oleh Gambar 5.14 baris 308.

```
282  public void onClick(
283      DialogInterface dialog,
284      int id) {
285      // get user input and set it to
286      // result
287      if (jumlah.getText()
288          .toString().trim()
289          .equals("")) {
290          jumlah.setError("Field is required!");
291      } else if (content.getText()
292          .toString().trim()
293          .equals("")) {
294          content.setError("Field is required!");
295      } else {
296          Catalog c = catalogs
297              .getItemById(idCatalog
298                  .get(position));
299          OrderItem o = new OrderItem();
300          o.setItem(c);
301          o.setAmount(Long
302              .parseLong(jumlah
303                  .getText()
304                  .toString()));
305          o.setContent(content
306              .getText()
307              .toString());
308          Keranjang
309              .addOrderBelanja(o);
310          Toast.makeText(
311              getApplicationContext(),
312              "Berhasil dimasukkan keranjang",
313              Toast.LENGTH_SHORT)
314          .show();
315      }
316 }
```

Gambar 5.14 Kode fungsi onClick pada AlertDialog

- **Tampilan activity_keranjang_belanja.xml**

Pada tampilan ini terdapat kontrol *ListView* untuk menampilkan *list* keranjang belanja yang dimiliki

pengguna. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.12.

```
36    <ListView  
37        android:id="@+id/ListKeranjang"  
38        android:layout_width="match_parent"  
39        android:layout_height="300dp" >  
40    </ListView>
```

Gambar 5.15 Kode program ActivityKeranjangBelanja.xml

- **Kelas KeranjangFragment.java**

- a. **Fungsi onCreateView**

Fungsi ini digunakan untuk menampilkan keranjang belanja yang telah dimiliki pengguna. Kemudian seluruh keranjang belanja pengguna yang telah disimpan sementara di kelas Keranjang tersebut ditampilkan dengan kelas KeranjangFragment menggunakan kontrol *ListView*. Inti fungsi ini ditunjukkan oleh Gambar 5.16 baris 86.

- b. **Fungsi hitungTotal()**

Fungsi ini digunakan untuk menghitung seluruh total keranjang belanja dengan cara menambahkan seluruh bunga yang dipesan. Inti fungsi ini ditunjukkan oleh Gambar 5.17 baris 138.

5.1.4. Memperbarui profil pengguna

- **Kelas AkunSayaFragment.java**

Pada kelas ini terdapat fungsi *onCreateView()* untuk menampilkan data profil pengguna dengan menggunakan data *session* pengguna. *Session* yang dimiliki pengguna melalui kelas *Credential*, akan disimpan di kelas *Constant* ketika pengguna melakukan proses *login* pada aplikasi. Fungsi ini ditunjukkan oleh Gambar 5.18.

```
67     public View onCreateView(LayoutInflater inflater, ViewGroup container,
68         Bundle savedInstanceState) {
69         View rootView = inflater.inflate(R.layout.activity_keranjang_belanja,
70             container, false);
71
72         imageLoader = ImageLoader.getInstance();
73         imageLoader.init(ImageLoaderConfiguration.createDefault(context));
74
75         options = new DisplayImageOptions.Builder()
76             .showImageForEmptyUri(R.drawable.ic_empty)
77             .showImageOnFail(R.drawable.ic_error)
78             .resetViewBeforeLoading(true).cacheOnDisc(true)
79             .imageScaleType(ImageScaleType.EXACTLY)
80             .bitmapConfig(Bitmap.Config.RGB_565).considerExifParams(true)
81             .displayer(new FadeInBitmapDisplayer(300)).build();
82
83         listView = (ListView) rootView.findViewById(R.id.listViewKeranjang);
84
85         // listView
86         CustomBaseAdapter adapter = new CustomBaseAdapter(context,
87             Keranjang.getAllKeranjang());
88         listView.setAdapter(adapter);
89
90         return rootView;
91     }
```

Gambar 5.16 Kode fungsi onCreateView

```
135     public Long hitungTotal() {
136         Long sum = (long) 0;
137         for (int i = 0; i < Keranjang.sizeKeranjang(); i++) {
138             sum += Keranjang.getAllKeranjang().get(i).getAmount()
139                 * Keranjang.getAllKeranjang().get(i).getItem().getPrice();
140         }
141
142         Log.d("SUM", sum.toString());
143         Credential.saveCredentialDataLong(context, BaseID.KEY_TOTAL_BELANJA,
144             sum);
145         return sum;
146     }
```

Gambar 5.17 Kode fungsi hitungTotal

```
38     public View onCreateView(LayoutInflater inflater, ViewGroup container,
39         Bundle savedInstanceState) {
40             View rootView = inflater.inflate(R.layout.activity_akun_saya,
41                 container, false);
42             Log.d("SESSION", Constant.session.toString());
43             if (Constant.session != null) {
44                 TextView text_namaUser = (TextView) rootView
45                     .findViewById(R.id.text_namaUser);
46                 TextView text_emailUser = (TextView) rootView
47                     .findViewById(R.id.text_emailUser);
48                 TextView text_telpUser = (TextView) rootView
49                     .findViewById(R.id.text_telpUser);
50                 TextView text_alamatUser = (TextView) rootView
51                     .findViewById(R.id.text_alamatUser);
52                 TextView text_notifUser = (TextView) rootView
53                     .findViewById(R.id.text_notifUser);
54
55             try {
56                 text_namaUser
57                     .setText(Constant.session.getFullscreen().toString());
58                 text_emailUser.setText(Constant.session.getEmail().toString());
59                 text_telpUser.setText(Constant.session.getPhone().toString());
60                 text_alamatUser.setText(Constant.session.getAddress()
61                     .toString());
62                 boolean isNotifOn = Credential.getBooleanCredentialData(getActivity(), BaseID.KEY_NOTIF_ON);
63                 if(isNotifOn){
64                     text_notifUser.setText("ON");
65                 } else{
66                     text_notifUser.setText("OFF");
67                 }
68             return rootView;
69         }
70     }
```

Gambar 5.18 Kode fungsi onCreateView

- **Kelas EditAkunSaya.java**

Pada kelas ini terdapat fungsi doInBackground() untuk menyimpan perubahan data profil pengguna. Inti fungsi ini ditunjukkan oleh Gambar 5.19 baris 163.

5.1.5. Melihat histori pemesanan dan detail histori pemesanan

- **Tampilan fragment_histori_pemesanan.xml**

Pada tampilan ini terdapat kontrol *ListView* untuk menampilkan *list* histori pemesanan yang dimiliki pengguna berdasarkan tanggal. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.20.

- **Kelas HistoriPemesananFragment.java**

Pada kelas ini terdapat fungsi doInBackground() untuk melakukan *query* di *background* ketika halaman dimuat. Fungsi ini melakukan *query* histori pemesanan

yang pernah dimiliki oleh pengguna. Inti fungsi ini ditunjukkan oleh Gambar 5.21 baris 101.

```
153@    protected User doInBackground(User... user) {  
154        // android.os.Debug.waitForDebugger();  
155        User userReturn = null;  
156        try {  
157            Log.d("Try", "Masuk");  
158            FloristAPI.Builder builder = new FloristAPI.Builder(  
159                AndroidHttp.newCompatibleTransport(),  
160                new AndroidJsonFactory(), null);  
161            builder.setRootUrl(Constant.URL);  
162            FloristAPI service = builder.build();  
163            userReturn = service.updateUser(user[0]).execute();  
164            // ifLog.d("Size", String.valueOf(userReturn));  
165        } catch (Exception e) {  
166            Log.d("Could not retrieve Catalog", e.getMessage(), e);  
167        }  
168        return userReturn;  
169    }  
170}
```

Gambar 5.19 Kode fungsi doInBackground asynctask EditAkunSaya

```
30    <ListView  
31        android:id="@+id/listViewTanggal"  
32        android:layout_width="match_parent"  
33        android:layout_height="match_parent"  
34        android:layout_below="@+id/textViewJudul"  
35        android:listSelector="@drawable/bg_selector_pink" >  
36    </ListView>
```

Gambar 5.20 Kode program fragment_histori_pemesanan.xml

- **Tampilan activity_histori_pemesanan_listview.xml**
Pada tampilan ini terdapat kontrol *ListView* untuk menampilkan *list* detail histori pemesanan yang berisi *list* bunga yang pernah dipesan pada tanggal histori pemesanan yang dipilih pengguna. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.22.

```
90     protected List<Order> doInBackground(Long... param) {  
91         // android.os.Debug.waitForDebugger();  
92         // Order orderReturn = null;  
93  
94         try {  
95             Log.d("Try", "Masuk");  
96             FloristAPI.Builder builder = new FloristAPI.Builder(  
97                 AndroidHttp.newCompatibleTransport(),  
98                 new AndroidJsonFactory(), null);  
99             builder.setRootUrl(Constant.URL);  
100            FloristAPI service = builder.build();  
101            listOrders = service.historyOrders(param[0]).execute()  
102                .getItems();  
103                // ifLog.d("Size", String.valueOf(userReturn));  
104            } catch (Exception e) {  
105                Log.d("Could not retrieve Catalog", e.getMessage(), e);  
106            }  
107        return listOrders;  
108    }
```

Gambar 5.21 Kode fungsi doInBackground asynctask HistoriPemesananFragmant

```
33     <ListView  
34         android:id="@+id/listViewPemesanan"  
35         android:layout_width="wrap_content"  
36         android:layout_height="wrap_content"  
37         android:layout_below="@+id/Layout"  
38     </ListView>
```

Gambar 5.22 Kode program activity_histori_pemesanan_listview.xml

- **Kelas HistoriPemesananActivity.java**

Pada kelas ini terdapat fungsi onCreate untuk menampilkan hasil *query* yang sudah dilakukan oleh fungsi pada gambar Gambar 5.21. Fungsi ini menampilkan detil histori pemesanan bunga dengan kontrol ListView. Inti fungsi ini ditunjukkan oleh Gambar 5.23 baris 76.

```
56     protected void onCreate(Bundle savedInstanceState) {
57         super.onCreate(savedInstanceState);
58         setContentView(R.layout.activity_histori_pemesanan_listview);
59
60         imageLoader = ImageLoader.getInstance();
61         imageLoader.init(ImageLoaderConfiguration.createDefault(this));
62
63         options = new DisplayImageOptions.Builder()
64             .showImageForEmptyUri(R.drawable.ic_empty)
65             .showImageOnFail(R.drawable.ic_error)
66             .resetViewBeforeLoading(true).cacheOnDisc(true)
67             .imageScaleType(ImageScaleType.EXACTLY)
68             .bitmapConfig(Bitmap.Config.RGB_565).considerExifParams(true)
69             .displayer(new FadeInBitmapDisplayer(300)).build();
70
71         listViewPemesanan = (ListView) findViewById(R.id.listViewPemesanan);
72
73         Intent i = getIntent();
74         pos = i.getIntExtra("position", 0);
75
76         CustomPemesananAdapter adapter = new CustomPemesananAdapter(getApplicationContext(), Constant.ListOrder.get(pos).getItems());
77         listViewPemesanan.setAdapter(adapter);
78         Credential.saveBooleanCredentialData(getApplicationContext(), BaseID.KEY_LOGIN, true);
79     }
```

Gambar 5.23 Kode fungsi onCreate

5.1.6. Menampilkan histori notifikasi

- **Tampilan fragment_push_notif.xml**

Pada tampilan ini terdapat kontrol *ListView* untuk menampilkan *list* notifikasi yang pernah didapatkan oleh pengguna. Kode sumber tampilan yang dimaksud ditunjukkan oleh Gambar 5.24.

```
30     <ListView
31         android:id="@+id/listViewNotif"
32         android:layout_width="match_parent"
33         android:layout_height="match_parent"
34         android:layout_below="@+id/textViewJudul">
35     </ListView>
```

Gambar 5.24 Kode program fragmetn_push_notif.xml

- **Kelas HistoriNotifikasiFragment.java**

Pada kelas ini terdapat fungsi *doInBackground()* untuk melakukan *query* terhadap seluruh notifikasi yang pernah didapatkan oleh pengguna. Inti fungsi ini ditunjukkan oleh Gambar 5.25 baris 93.

```

82    protected List<Notification> doInBackground(Long... param) {
83        // android.os.Debug.waitForDebugger();
84        // Order orderReturn = null;
85
86        try {
87            Log.d("Try", "Masuk");
88            FloristAPI.Builder builder = new FloristAPI.Builder(
89                AndroidHttp.newCompatibleTransport(),
90                new AndroidJsonFactory(), null);
91            builder.setRootUrl(Constant.URL);
92            FloristAPI service = builder.build();
93            listNotif = service.historyNotifications(param[0]).execute()
94                .getItems();
95            // ifLog.d("Size", String.valueOf(userReturn));
96        } catch (Exception e) {
97            Log.d("Could not retrieve Catalog", e.getMessage(), e);
98        }
99        return listNotif;
100    }

```

Gambar 5.25 Kode fungsi doInBackground asynctask HistoriNotifikasiFrgment

5.1.7. Melakukan registrasi

- **Kelas RegisterActivity.java**

Pada kelas ini terdapat fungsi doInBackground() untuk melakukan penyimpanan data registrasi pengguna. Inti fungsi ini ditunjukkan oleh Gambar 5.26 baris 132.

```

122    protected User doInBackground(User... user) {
123        // android.os.Debug.waitForDebugger();
124        User userReturn = null;
125        try {
126            Log.d("Try", "Masuk");
127            FloristAPI.Builder builder = new FloristAPI.Builder(
128                AndroidHttp.newCompatibleTransport(),
129                new AndroidJsonFactory(), null);
130            builder.setRootUrl(Constant.URL);
131            FloristAPI service = builder.build();
132            userReturn = service.registerUser(password, user[0]).execute();
133            // ifLog.d("Size", String.valueOf(userReturn));
134        } catch (Exception e) {
135            Log.d("Could not retrieve Catalog", e.getMessage(), e);
136        }
137        return userReturn;
138    }

```

Gambar 5.26 Kode fungsi doInBackground asynctask RegisterActivity

5.1.8. Melakukan pemesanan bunga

- **Kelas OrderActivity.java**

Pada kelas ini terdapat fungsi doInBackground() untuk melakukan penyimpanan data pemesanan bunga oleh pengguna. Inti fungsi ini ditunjukkan oleh Gambar 5.27 baris 133.

```
125@      protected Boolean doInBackground(Order... param) {  
126        try {  
127          Log.d("Try", "Masuk");  
128          FloristAPI.Builder builder = new FloristAPI.Builder(  
129              AndroidHttp.newCompatibleTransport(),  
130              new AndroidJsonFactory(), null);  
131          builder.setRootUrl(Constant.URL);  
132          FloristAPI service = builder.build();  
133          service.addOrder(param[0]).execute();  
134          return true;  
135          // ifLog.d("Size", String.valueOf(userReturn));  
136        } catch (Exception e) {  
137          Log.d("Could not retrieve Catalog", e.getMessage(), e);  
138          return false;  
139        }  
140      }
```

Gambar 5.27 Kode fungsi doInBackground asynctask OrderActivity

5.1.9. Melakukan permintaan *reset password*

- **Kelas ForgotPassword.java**

Pada kelas ini terdapat fungsi doInBackground() untuk melakukan permintaan *reset password* jika pengguna ingat dengan *email* namun lupa dengan *password* akunnya. Setelah mengeksekusi fungsi ini, pengguna akan mendapatkan *email* untuk melakukan pergantian *password* dengan yang baru. Inti fungsi ini ditunjukkan oleh Gambar 5.28 baris 83.

```
74@          protected Void doInBackground(String... param) {  
75      // android.os.Debug.waitForDebugger();  
76      try {  
77          Log.d("Try", "Masuk");  
78          FloristAPI.Builder builder = new FloristAPI.Builder(  
79              AndroidHttp.newCompatibleTransport(),  
80              new AndroidJsonFactory(), null);  
81          builder.setRootUrl(Constant.URL);  
82          FloristAPI service = builder.build();  
83          service.forgotPassword(param[0]).execute();  
84          // ifLog.d("Size", String.valueOf(userReturn));  
85      } catch (Exception e) {  
86          Log.d("Could not retrieve Catalog", e.getMessage(), e);  
87      }  
88      return null;  
89  }
```

**Gambar 5.28 Kode fungsi doInBackground asynctask
ForgotPassword**

5.1.10. Menerima *push notification* dari Meme Florist

- **Kelas GCMIntentService.java**

6.1 Fungsi registeredAtGCM(Context context)

Pada kelas ini terdapat fungsi registeredAtGCM() yang digunakan untuk melakukan pemeriksaan apakah *device* yang digunakan pengguna telah memiliki registrasi GCM ID. Pemanggilan fungsi ini berada di kelas Splashscreen, karena kelas Splashscreen adalah kelas yang pertama kali dieksekusi ketika awal aplikasi dijalankan. Jika *device* belum terdaftar, maka secara otomatis GCM akan menjalankan fungsinya dalam mendaftarkan *device* tersebut untuk bisa mendapatkan registrasi GCM ID. Fungsi ini ditunjukkan oleh Gambar 5.29.

```
103 public static void registerAtGCM(Context context) {  
104     GCMRegistrar.checkDevice(context);  
105     GCMRegistrar.checkManifest(context);  
106     String regId = GCMRegistrar.getRegistrationId(context);  
107     Log.d("regID", regId);  
108     if (regId.equals("")) {  
109         GCMRegistrar.register(context, SENDER_ID);  
110         Credential.saveBooleanCredentialData(context, BaseID.KEY_NOTIF_ON, true);  
111         Log.d("id", "kosong");  
112     } else {  
113         Log.d(TAG, "Already registered: " + regId);  
114         registerGCMClient(context, regId);  
115     }  
116 }  
117 }
```

Gambar 5.29 Kode fungsi registerAtGCM

6.2 Fungsi showNotification(String title, String message, String url, String idShipment)

Pada kelas ini terdapat fungsi showNotification yang digunakan untuk menampilkan notifikasi yang berasal dari GCM. Notifikasi yang diberikan oleh GCM akan masuk ke dalam *notification bar device*, dan notifikasi tersebut ketika dilakukan *action* oleh pengguna misalnya ditekan, maka dapat diolah untuk bisa diarahkan membuka aplikasi secara otomatis dan memindahkan antarmuka ke fitur tertentu di dalam aplikasi. Fungsi ini ditunjukkan oleh Gambar 5.30.

5.1.11. Melakukan pengaturan penerimaan *push notification*

- **Kelas GCMIntentService.java**

Pada kelas ini terdapat fungsi registerGCMClient yang digunakan untuk melakukan pengaturan terhadap pengiriman notifikasi *broadcast* untuk *device* tertentu. Pengguna dapat melakukan pengaturan nonaktif notifikasi yang berada di dalam fitur edit akun kemudian menonaktifkan *toogle* yang ada. Begitu juga sebaliknya, dan perubahan pengaturan tersebut disimpan saat pengguna memiliki *session* berupa kelas

Credential. Data session Credential ini akan diinisialisasikan fungsi registerGCMClient untuk dapat melakukan pengaturan notifikasi *true* atau *false* berdasarkan *session* Credential yang dimiliki pengguna. Fungsi ini ditunjukkan oleh Gambar 5.30.

```

51@ private void showNotification(String title, String message, String url, String idShipment) {
52     NotificationCompat.Builder mBuilder =
53         new NotificationCompat.Builder(this)
54             .setSmallIcon(R.drawable.ic_launcher)
55             .setContentTitle(title)
56             .setContentText(message)
57             .setAutoCancel(true);
58     // Creates an explicit intent for an Activity in your app
59     Intent resultIntent = new Intent(this, MainActivity.class);
60     resultIntent.putExtra("title", title);
61     resultIntent.putExtra("message", message);
62     resultIntent.putExtra("url", url);
63     resultIntent.putExtra("idshipment", idShipment);
64
65     // The stack builder object will contain an artificial back stack for
66     // the
67     // started Activity.
68     // This ensures that navigating backward from the Activity leads out of
69     // your application to the Home screen.
70     TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
71     // Adds the back stack for the Intent (but not the Intent itself)
72     stackBuilder.addParentStack(Splashscreen.class);
73     // Adds the Intent that starts the Activity to the top of the stack
74     stackBuilder.addNextIntent(resultIntent);
75     PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0,
76         PendingIntent.FLAG_UPDATE_CURRENT);
77     mBuilder.setContentIntent(resultPendingIntent);
78     NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
79     // mId allows you to update the notification later on.
80     mNotificationManager.notify(1, mBuilder.build());
81 }
```

Gambar 5.30 Kode fungsi showNotification

```

119@ public static void registerGCMClient(Context context, String registrationId) {
120
121     GCMDevices gcm = new GCMDevices();
122     if (Constant.session != null) {
123         gcm.setAcceptPersonalNotification(true);
124         Log.d("session", Constant.session.toString());
125         gcm.setUser(Constant.session);
126     }
127     gcm.setDeviceRegId(registrationId);
128     boolean isNotifOn = Credential.getBooleanCredentialData(context, BaseID.KEY_NOTIF_ON);
129     gcm.setAcceptBroadcastNotification(isNotifOn);
130
131     new Registration(context).execute(gcm);
132 }
```

Gambar 5.31 Kode fungsi registerGCMClient

5.2 Implementasi Website Administrator Meme Florist

Di bagian ini akan dijelaskan mengenai implementasi berupa kode program yang digunakan untuk membangun aplikasi *web* Meme Florist. Kode program telah terbagi menjadi beberapa bagian baik untuk tampilan di klien (.vm) maupun kode di belakangnya (*code behind*) dalam bahasa pemrograman Java. Di dalam tiap-tiap kode program tersebut akan dijelaskan dengan keterangan tentang maksud kodennya. Secara detail mengenai implementasi aplikasi *web* Meme Florist dijabarkan sebagai berikut:

5.2.1. Login Administrator Meme Florist

- **Tampilan login.vm**

Pada tampilan *login* administrator ini dibutuhkan kontrol *Input type Text* untuk pengisian *username*, kontrol *Input type Password* untuk pengisian *password* dan kontrol *Button* yang bertuliskan “Login” untuk *trigger* pengiriman *login form*. Kode lain dalam tampilan login.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.32

```
<div class="container">
    <form class="form-signin" role="form" action="login" method="post">
        <div align="center">
            
            <h2 class="form-signin-heading">Login Administrator</h2>
            <input type="text" class="form-control" placeholder="Username"
                   name="username" required autofocus> <input type="password"
                   class="form-control" placeholder="Password" name="password" required>
            <button class="btn btn-lg btn-primary btn-block" type="submit"
                   style="margin-left: 0px">Login</button>
        </div>
    </form>
</div>
```

Gambar 5.32 Kode tampilan login.vm

- **Kelas LoginServlet.java**

Kelas ini merupakan *code behind* dari tampilan login.vm. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Di bagian bawah potongan kode program ini, hanya melakukan mekanisme pemanggilan *Velocity Template* dengan tampilan dari login.vm dan penggabungan antara *template* dengan *Velocity Context* oleh *Velocity Engine*. Sehingga tampilan yang terlihat adalah sama seperti tampilan yang ada di dalam login.vm. Fungsi ini ditunjukkan oleh Gambar 5.33.

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
Constant.Login(response);
VelocityContext context = new VelocityContext();
context.put("baseUrl", Constant.baseUrl);
VelocityEngine ve = VelocityHelper.getVelocityEngine();
Template template = ve.getTemplate("login.vm");
StringWriter writer = new StringWriter();
template.merge(context, writer);
response.setContentType("text/html");
response.getWriter().print(writer.toString());
}
```

Gambar 5.33 Kode fungsi doGet LoginServlet

b. Fungsi doPost(HttpServletRequest request, HttpServletResponse response)

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Kemudian fungsi ini akan mengambil *value* dari *request* yang dihasilkan oleh *form* berupa *username* dan *password*. Selanjutnya adalah melakukan *query* pengecekan berupa fungsi *loginCompany* dengan menggunakan kelas *FloristEndpoint*. Jika nilai pengembalian bukan berupa *null*, maka *login* sukses dan *set session* untuk konstanta *company* dan diarahkan menuju

halaman beranda, selain itu maka *login* gagal dan diarahkan kembali ke halaman *login*. Fungsi ini ditunjukkan oleh Gambar 5.34.

```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    Constant.login(response);
    String password = request.getParameter("password");
    String username = request.getParameter("username");
    FloristEndpoint api = new FloristEndpoint();
    Company company = api.loginCompany(username, password);
    if (company != null) {
        Constant.company = company;
        response.sendRedirect("/dashboard");
    } else {
        doGet(request, response);
    }
}
```

Gambar 5.34 Kode fungsi doPost LoginServlet

5.2.2. Menampilkan Halaman Beranda Meme Florist

- **Tampilan dashboard.vm**

Pada tampilan halaman beranda ini tidak terdapat banyak kontrol karena yang memiliki kontrol adalah *template* yang dimasukkan (*include*) ke dalam tampilan halaman beranda ini, yakni header.vm dan navigation.vm. Kode lain tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.35.

```
#parse ( "header.vm" )
<div class="container-fluid">
<div class="row">
    #parse ( "navigation.vm" )
    <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">
        <h1 class="page-header">Beranda</h1>
        <div class="row">
            <div class="col-md-12" align="center">
                
            </div>
        </div>
    </div>
</div>
```

Gambar 5.35 Kode tampilan dashboard.vm

- **Kelas DashboardServlet.java**

Kelas ini merupakan *code behind* dari tampilan dashboard.vm. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai *administrator*. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Di bagian bawah potongan kode program ini, hanya melakukan mekanisme pemanggilan *Velocity Template* dengan tampilan dari login.vm dan penggabungan antara *template* dengan *Velocity Context* oleh *Velocity Engine*. Sehingga tampilan yang terlihat adalah sama seperti tampilan yang ada di dalam dashboard.vm. Fungsi ini ditunjukkan oleh Gambar 5.36.

- b. **Fungsi doPost(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*. Karena tidak ada *form* yang melakukan *submit* dengan metode *post* ke

kelas DashboardServlet maka di dalam fungsi ini hanya diarahkan untuk memanggil fungsi doGet. Fungsi ini ditunjukkan oleh Gambar 5.37.

```
protected void doGet(HttpServletRequest request,  
                     HttpServletResponse response) throws ServletException, IOException {  
    Constant.checkSession(response);  
    FloristEndpoint api = new FloristEndpoint();  
    VelocityContext context = new VelocityContext();  
    context.put("baseUrl", Constant.baseUrl);  
    context.put("page", "dashboard");  
    VelocityEngine ve = VelocityHelper.getVelocityEngine();  
    Template template = ve.getTemplate("dashboard.vm");  
    StringWriter writer = new StringWriter();  
    template.merge(context, writer);  
    response.setContentType("text/html");  
    response.getWriter().print(writer.toString());  
}
```

Gambar 5.36 Kode fungsi doGet DashboardServlet

```
protected void doPost(HttpServletRequest request,  
                     HttpServletResponse response) throws ServletException, IOException {  
    doGet(request, response);  
}
```

Gambar 5.37 Kode fungsi doPost DashboardServlet

5.2.3. Melihat Daftar Katalog Bunga

- **Tampilan catalog.vm**

Pada tampilan melihat daftar katalog bunga ini terdapat beberapa kontrol, beberapa diantaranya yaitu untuk fitur *filter* kategori yang menggunakan kontrol *Select*, tombol Tambah baru, Sunting dan Hapus yang menggunakan kontrol *Button*. Isi dari *filter* kategori dan semua *item* katalog yang ditampilkan diisi oleh *code behind* yang ada di CatalogServlet.java. Kode lain dalam tampilan catalog.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.38.

```
#set( $iter = 1 )
#foreach($i in $catalogs)
<div class="col-sm-6 col-md-4">
  <div class="thumbnail" style="background-color: #124347; color: #FFF">
    <div align="center">
      <table class="table table-condensed" style="margin-bottom: 0px">
        <tr>
          <td rowspan="5">
            <div style="max-width: 100%; max-height: 100px; overflow: hidden"
                align="center">
              <a href="$baseUrl/serve?blob-key=$i.imageId" class="preview"
                 title="$i.name"></a>
            </div>
          </td>
          <td class="info" align="center" style="color: #000"><b>$i.name</b></td>
        </tr>
        <tr>
          <td align="center">Rp. $i.price,-</td>
        </tr>
        <tr>
          <td align="center">$i.width cm x $i.height cm</td>
        </tr>
        <tr>
          <td align="center">$i.category.name</td>
        </tr>
        <tr>
          <td align="center"><button type="button"
              onclick="window.location.href = '$baseUrl/catalog/update?idCatalog='+ $i.id"
              class="btn btn-primary btn-sm">
            <span class="glyphicon glyphicon-edit"></span> Sunting
          </button>
          <button type="button" onclick="removeCatalog('$i.id','$i.name')"
              class="btn btn-default btn-sm">
            <span class="glyphicon glyphicon-trash"></span> Hapus
          </button></td>
        </tr>
      </table>
    </div>
  </div>
</div>
#end
```

Gambar 5.38 Kode tampilan catalog.vm (*view*)

- **Kelas CatalogServlet.java**

Kelas ini merupakan *code behind* dari tampilan catalog.vm untuk *view* (melihat) dan *delete* (menghapus), namun yang dibahas kali ini adalah *view*. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis

HTTP request berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai *administrator*. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Di dalam fungsi ini terdapat variabel *removeId* untuk kasus penggunaan menghapus katalog, namun untuk melihat katalog, variabel ini bernilai *null*. Kemudian dengan menggunakan fungsi *getCategories* dan fungsi *getCatalogs* dari kelas *FloristEndpoint*, maka didapatkan kategori dan katalog yang diperlukan. Setelah itu dilakukan *merging* sebagai *context* dengan *template* untuk ditampilkan oleh *Velocity Engine*. Kode lain dalam fungsi *doGet* ini tidak ditampilkan dalam bagian laporan ini. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.39.

5.2.4. Menambah Data Katalog Bunga

- **Tampilan catalog.vm**

Pada tampilan menambah data katalog bunga ini terdapat beberapa kontrol, beberapa diantaranya yaitu untuk isian kategori yang menggunakan kontrol *Select*, isian Nama, Lebar, Tinggi, Deskripsi menggunakan kontrol *Input type Text*, isian Harga menggunakan kontrol *Input type Number*, isian Gambar menggunakan kontrol *Input type File*, tombol Simpan dan Batal yang

menggunakan kontrol *Button*. Isi dari kategori diisi oleh *code behind* yang ada di CatalogAddServlet.java. Kode lain dalam tampilan catalog.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.40.

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    String removeId = request.getParameter("removeIdCatalog");
    if (removeId == null) {
        VelocityContext context = new VelocityContext();
        VelocityEngine ve = VelocityHelper.getVelocityEngine();
        Template template = ve.getTemplate("catalog.vm");
        StringWriter writer = new StringWriter();

        List<Category> list_cat = api.getCategories(null);

        String catFilter = request.getParameter("catId");
        List<Catalog> res;
        if (catFilter != null && !catFilter.equals("-1")) {
            res = api.getCatalogsByCategory(Long.valueOf(catFilter));
        } else {
            res = api.getCatalogs(null);
        }
        context.put("baseUrl", Constant.baseUrl);
        context.put("param", "");
        context.put("catalogs", res);
        context.put("page", "catalog");
        context.put("list_cat", list_cat);
        template.merge(context, writer);
        response.setContentType("text/html");
        response.getWriter().print(writer.toString());
    } else {
        **
    }
}
```

Gambar 5.39 Kode fungsi doGet CatalogServlet (*view*)

```
#if ( $param=="add" )
<div style="width: 50%">
  <form role="form" action="$blobstoreURL" method="post"
    enctype="multipart/form-data">
    <table class="table table-striped table-condensed">
      <tr>
        <td><b>Kategori</b></td>
        <td colspan="2"><select class="form-control" id="category"
          name="category">
          #foreach($cat in $list_cat)
            <option value="$cat.id">$cat.name</option>
          #end
        </select></td>
      </tr>
      <tr>
        <td><b>Nama</b></td>
        <td colspan="2"><input class="form-control" type="text"
          id="name" name="name" required></td>
      </tr>
      ***
      <tr>
        <td></td>
        <td>
          <button type="submit" class="btn btn-primary btn-block">
            <span class="glyphicon glyphicon-floppy-save"></span> Simpan
          </button>
        </td>
        <td>
          <button onclick="window.location.href='$baseUrl/catalog'">
            class="btn btn-default btn-block">
              <span class="glyphicon glyphicon-floppy-remove"></span> Batal
          </button>
        </td>
      </tr>
    </table>
  </form>
</div>
#elseif ( $param=="update" )
-
#else
-
#end
```

Gambar 5.40 Kode tampilan catalog.vm (add)

- **Kelas CatalogAddServlet.java**

Kelas ini merupakan *code behind* dari tampilan catalog.vm untuk add (menambah). Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

a. Fungsi doGet(HttpServletRequest request, HttpServletResponse response)

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai *administrator*. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Kemudian dengan menggunakan fungsi *getCategories* dari kelas *FloristEndpoint*, maka didapatkan kategori yang diperlukan. Selanjutnya karena terdapat mekanisme *upload file* berupa gambar maka dibutuhkan pembubuhan *uploadUrl* yang diperoleh dari *BlobstoreService*. Setelah itu dilakukan *merging* sebagai *context* dengan *template* untuk ditampilkan oleh *Velocity Engine*. Kode lain dalam fungsi *doGet* ini tidak ditampilkan dalam bagian laporan ini. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.41.

b. Fungsi doPost(HttpServletRequest request, HttpServletResponse response)

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*.

Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Kemudian fungsi ini akan mengambil *value* dari *request* yang dihasilkan oleh *form* berupa *category*, *name*, *width*, *height*, *price*, dan *desc* serta dilakukan penyimpanan gambar yang dilakukan oleh *BlobstoreService*. Selanjutnya adalah melakukan *insert* dengan menggunakan fungsi *addCatalog* dari kelas *FloristEndpoint*. Setelah selesai lalu diarahkan kembali ke halaman melihat katalog. Fungsi ini ditunjukkan oleh Gambar 5.42.

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    List<Category> list_cat = api.getCategories(null);
    VelocityContext context = new VelocityContext();
    VelocityEngine ve = VelocityHelper.getVelocityEngine();
    Template template = ve.getTemplate("catalog.vm");
    StringWriter writer = new StringWriter();
    BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
    String blobstoreURL = bs.createUploadUrl("/catalog/add");
    context.put("blobstoreURL", blobstoreURL);
    context.put("list_cat", list_cat);
    context.put("baseUrl", Constant.baseUrl);
    context.put("param", "add");
    template.merge(context, writer);
    response.setContentType("text/html");
    response.getWriter().print(writer.toString());
}
```

Gambar 5.41 Kode fungsi doGet CatalogAddServlet

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    Long idCategory = Long.valueOf(request.getParameter("category"));
    String name = request.getParameter("name");
    String width = request.getParameter("width");
    String height = request.getParameter("height");
    Long price = Long.valueOf(request.getParameter("price"));
    String desc = request.getParameter("desc");

    Category ca = api.getCategoryById(idCategory);

    BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
    Map<String, BlobKey> blobs = bs.getUploadedBlobs(request);
    BlobKey blobKey = blobs.get("image");
    String imageId = blobKey.getKeyString();

    Catalog c = new Catalog(name, width, height, price, ca, desc, imageId);
    api.addCatalog(c);
    response.sendRedirect("/catalog");
}
```

Gambar 5.42 Kode fungsi doPost CatalogAddServlet

5.2.5. Mengubah Data Katalog Bunga

- **Tampilan catalog.vm**

Pada tampilan menambah data katalog bunga ini terdapat beberapa kontrol, beberapa diantaranya yaitu untuk isian Nama, Lebar, Tinggi, Deskripsi menggunakan kontrol *Input type Text*, isian Harga menggunakan kontrol *Input type Number*, isian Gambar menggunakan kontrol *Input type File*, tombol *Update* dan *Batal* yang menggunakan kontrol *Button*. Isi dari masing-masing *input* diisi oleh *code behind* yang ada di CatalogUpdateServlet.java. Kode lain dalam tampilan catalog.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.43.

```
#if( $param=="add" )
-
#elseif ( $param=="update" )
<div style="width: 50%">
    <form role="form" action="$blobstoreURL" method="post"
        enctype="multipart/form-data">
        <input type="hidden" name="idCatalog" value="$catalog.id" />
        <table class="table table-striped table-condensed">
            <tr>
                <td width="110"><b>Nama</b></td>
                <td colspan="2"><input class="form-control" type="text"
                    id="name" name="name" value="$catalog.name" required></td>
            </tr>
            ●●●
            <tr>
                <td></td>
                <td>
                    <button type="submit" class="btn btn-primary btn-block">
                        <span class="glyphicon glyphicon-floppy-save"></span>Update
                    </button>
                </td>
                </tr>
                <td>
                    <button onclick="window.location.href='$baseUrl/catalog'"
                        class="btn btn-default btn-block">
                        <span class="glyphicon glyphicon-floppy-remove"></span> Batal
                    </button>
                </td>
            </tr>
        </table>
    </div>
#else
-
#end
```

Gambar 5.43 Kode tampilan catalog.vm (*update*)

- **Kelas CatalogUpdateServlet.java**

Kelas ini merupakan *code behind* dari tampilan catalog.vm untuk *update* (mengubah). Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan

request. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai *administrator*. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Kemudian fungsi ini akan mengambil *value* dari *request* dari metode *get* berupa *idCatalog*, yakni katalog yang akan diubah atau diperbarui datanya. Kemudian dengan menggunakan fungsi *getCatalogById* dari kelas *FloristEndpoint*, maka didapatkan katalog yang diminta. Selanjutnya karena terdapat mekanisme *upload file* berupa gambar maka dibutuhkan pembubuhan *uploadUrl* yang diperoleh dari *BlobstoreService*. Setelah itu dilakukan *merging* sebagai *context* dengan *template* untuk ditampilkan oleh *Velocity Engine*. Kode lain dalam fungsi *doGet* ini tidak ditampilkan dalam bagian laporan ini. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.44.

b. Fungsi *doPost(HttpServletRequest request, HttpServletResponse response)*

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam

keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Kemudian fungsi ini akan mengambil *value* dari *request* yang dihasilkan oleh *form* berupa *idCatalog*, *name*, *width*, *height*, *price*, dan *desc* serta dilakukan penyimpanan gambar yang dilakukan oleh *BlobstoreService*. Selanjutnya adalah melakukan *update* dengan menggunakan fungsi *updateCatalog* dari kelas *FloristEndpoint*. Setelah selesai lalu diarahkan kembali ke halaman melihat katalog. Fungsi ini ditunjukkan oleh Gambar 5.45 dan Gambar 5.46.

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    String idCatalog = request.getParameter("idCatalog");

    if (idCatalog != null) {
        FloristEndpoint api = new FloristEndpoint();
        Catalog c = api.getCatalogById(Long.valueOf(idCatalog));
        if (c != null) {
            VelocityContext context = new VelocityContext();
            VelocityEngine ve = VelocityHelper.getVelocityEngine();
            Template template = ve.getTemplate("catalog.vm");
            StringWriter writer = new StringWriter();
            BlobstoreService bs = BlobstoreServiceFactory
                .getBlobstoreService();
            String blobstoreURL = bs.createUploadUrl("/catalog/update");
            context.put("blobstoreURL", blobstoreURL);
            context.put("catalog", c);
            context.put("baseUrl", Constant.baseUrl);
            context.put("param", "update");
            template.merge(context, writer);
            response.setContentType("text/html");
            response.getWriter().print(writer.toString());
        } else {
            response.sendRedirect("/catalog");
        }
    } else
        response.sendRedirect("/catalog");
}
```

Gambar 5.44 Kode fungsi doGet CatalogUpdateServlet

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    String idCatalog = request.getParameter("idCatalog");
    String name = request.getParameter("name");
    String width = request.getParameter("width");
    String height = request.getParameter("height");
    Long price = Long.valueOf(request.getParameter("price"));
    String desc = request.getParameter("desc");
    Catalog c = api.getCatalogById(Long.valueOf(idCatalog));
    if (c != null) {
        BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
        Map<String, BlobKey> blobs = bs.getUploadedBlobs(request);
        BlobInfoFactory bif = new BlobInfoFactory();
        boolean contain = blobs.containsKey("image");
        if (contain) {
            BlobKey blobKey = blobs.get("image");
            BlobInfo blobInfo = bif.loadBlobInfo(blobKey);
            String imageId = "";
            if (blobInfo.getSize() != 0) {
                imageId = blobKey.getKeyString();
                c.setImageId(imageId);
            }
        }
    }
}
```

Gambar 5.45 Kode fungsi doPost CatalogUpdateServlet

5.2.6. Menghapus Data Katalog Bunga

- Tampilan catalog.vm

Pada tampilan menghapus data katalog bunga ini hanya membutuhkan *confirm dialog* yang telah disediakan oleh JavaScript. Untuk *request* yang diproses dalam menghapus data katalog terdapat pada *code behind* yang ada di CatalogServlet.java. Kode lain dalam tampilan catalog.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.47.

```
c.setName(name);
c.setWidth(width);
c.setHeight(height);
c.setDesc(desc);
c.setPrice(price);

        api.updateCatalog(c);
    }
response.sendRedirect("/catalog");
}
```

Gambar 5.46 Lanjutan kode fungsi doPost CatalogUpdateServlet

```
<script>
    function removeCatalog(id, name) {
        var r = confirm("Anda yakin untuk menghapus item " + name);
        if (r == true) {
            window.location.href = '$baseUrl/catalog?removeIdCatalog=' + id;
        }
    }
</script>
```

Gambar 5.47 Kode tampilan catalog.vm (delete)

- **Kelas CatalogServlet.java**

Kelas ini merupakan *code behind* dari tampilan catalog.vm untuk *view* (melihat) dan *delete* (menghapus), namun yang dibahas kali ini adalah *delete*. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain

ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai administrator. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Di dalam fungsi ini terdapat variabel *removeId* yang bernilai sesuai id katalog yang dipilih untuk dihapus. Kemudian dengan menggunakan fungsi *removeCatalogById* dari kelas *FloristEndpoint*, maka katalog yang dipilih telah dihapus. Setelah itu pengguna akan diarahkan kembali menuju halaman melihat katalog. Kode lain dalam fungsi *doGet* ini tidak ditampilkan dalam bagian laporan ini. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.48.

5.2.7. Melihat Daftar Pemesanan Pelanggan

- **Tampilan order.vm**

Pada tampilan melihat daftar pemesanan pelanggan ini terdapat beberapa kontrol, beberapa diantaranya yaitu untuk melihat daftar atribut dari pemesanan dengan menggunakan kontrol *Table*, dan tombol Konfirmasi yang menggunakan kontrol *Button*. Isi dari semua daftar pemesanan yang ditampilkan diisi oleh *code behind* yang ada di *OrderServlet.java*. Kode lain dalam tampilan *order.vm* ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.49.

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    String removeId = request.getParameter("removeIdCatalog");
    if (removeId == null) {
        /*
    } else {
        api.removeCatalogById(Long.valueOf(removeId));
        response.sendRedirect("/catalog");
    }
}
```

Gambar 5.48 Kode fungsi doGet CatalogServlet (delete)

- **Kelas OrderServlet.java**

Kelas ini merupakan *code behind* dari tampilan order.vm untuk melihat daftar pemesanan pelanggan. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai *administrator*. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Kemudian dengan menggunakan fungsi *getOrders* dari kelas *FloristEndpoint*, maka didapatkan daftar pemesanan yang diperlukan. Setelah itu dilakukan *merging* sebagai *context*

dengan *template* untuk ditampilkan oleh *Velocity Engine*. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.50.

5.2.8. Melakukan Konfirmasi Pemesanan Bunga

- **Tampilan order.vm**

Pada tampilan melakukan konfirmasi pemesanan bunga ini terdapat beberapa kontrol, beberapa diantaranya yaitu untuk melihat atribut dari detail pemesanan dengan menggunakan kontrol *Table*, isian untuk Nama Pemesan, Nama Penerima Barang, Email menggunakan kontrol *Input type Text*, isian untuk Catatan menggunakan kontrol *Textarea*, lampiran foto menggunakan *Input type File*, dan tombol Kirim dan Batal menggunakan kontrol *Button*. Isi dari semua detail pemesanan, nama pemesan dan *email* yang ditampilkan diisi oleh *code behind* yang ada di ConfirmServlet.java. Kode lain dalam tampilan order.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.51 dan Gambar 5.52.

- **Kelas ConfirmServlet.java**

Kelas ini merupakan *code behind* dari tampilan order.vm untuk melakukukan konfirmasi pemesanan bunga. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis

HTTP request berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai administrator. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Kemudian dengan menggunakan fungsi *getOrderById* dari kelas *FloristEndpoint*, maka didapatkan pemesanan yang akan dikonfirmasi. Selanjutnya karena terdapat mekanisme *upload file* berupa gambar maka dibutuhkan pembubuhan *uploadUrl* yang diperoleh dari *BlobstoreService*. Setelah itu dilakukan *merging* sebagai *context* dengan *template* untuk ditampilkan oleh *Velocity Engine*. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.53.

b. Fungsi `doPost(HttpServletRequest request, HttpServletResponse response)`

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam

keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Kemudian dilakukan penyimpanan gambar yang dilakukan oleh *BlobstoreService* serta menambahkan konfirmasi pemesanan ke dalam pengiriman dengan menggunakan fungsi *addShipment* dari kelas *FloristEndpoint*. Selain itu, terdapat mekanisme pengiriman *push notification* dan *email* kepada pelanggan. Setelah selesai lalu diarahkan kembali ke halaman melihat daftar pemesanan pelanggan. Kode lain dari fungsi ini tidak ditampilkan. Fungsi ini ditunjukkan oleh Gambar 5.54 dan Gambar 5.55.

5.2.9. Mengirim Pemberitahuan kepada Pelanggan

- Tampilan notification.vm**

Pada tampilan melakukan konfirmasi pemesanan bunga ini terdapat beberapa kontrol, beberapa diantaranya yaitu pemilihan *mode* dengan menggunakan kontrol *Select*, isian untuk Nama, Judul Pesan dan Isi Pesan menggunakan kontrol *Input type Text*, dan tombol Kirim menggunakan kontrol *Button*. Kode lain dalam tampilan notification.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.56.

```
#if ( $param=="confirm" )
...
#else
<style>
th {
    text-align: center;
}
</style>
<div class="table-responsive">
    <table class="table table-striped table-bordered">
        <thead>
            <tr>
                style="font-weight: bold; font-size: 14px; background-color: #3BC1C3; color: #FFF;">
                    <th>No</th>
                    <th>Nama Pemesan</th>
                    <th>Total Harga</th>
                    <th>Status</th>
                    <th>Alamat Tujuan</th>
                    <th>Konfirmasi</th>
                </tr>
        </thead>
        <tbody>
            #set( $iter = 1 )..
            #foreach($i in $orders)
            <tr>
                <td>$iter</td>..
                #set( $iter = $iter+1 )..
                <td>$i.sender.fullname</td>
                <td>$i.totalPrice</td>..
                #if($i.status)
                    <td><span class="glyphicon glyphicon-ok-sign"
                        style="color: #47c914;"></span> Sudah</td>..
                #else
                    <td><span class="glyphicon glyphicon-remove-sign"
                        style="color: #fd0202;"></span> Belum</td>..
                #end
                <td>$i.recipientAddress</td>
                <td><button type="button"
                    onclick="window.location.href = '$baseUrl/order/confirm?id=' + $i.id"
                    class="btn btn-info">
                        <span class="glyphicon glyphicon-check"></span> Konfirmasi
                    </button></td>
                </tr>
            #end
        </tbody>
    </table>
</div>
#end
```

Gambar 5.49 Kode tampilan order.vm (*view*)

```

protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);

    FloristEndpoint api = new FloristEndpoint();
    VelocityContext context = new VelocityContext();
    VelocityEngine ve = VelocityHelper.getVelocityEngine();
    Template template = ve.getTemplate("order.vm");
    StringWriter writer = new StringWriter();

    List<Order> list_order = api.getOrders(null);

    context.put("baseUrl", Constant.baseUrl);
    context.put("param", "");
    context.put("orders", list_order);
    context.put("page", "order");
    template.merge(context, writer);
    response.setContentType("text/html");
    response.getWriter().print(writer.toString());
}

```

Gambar 5.50 Kode fungsi doGet OrderServlet

```

@if ($param=="confirm")


## Detail



|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <b>Pengirim</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <a href="#">\$baseUrl/user/detail?Id=\$order.sender.id"&gt;\$order.sender.fullname&lt;/a&gt;</a> |
| <h2 class="sub-header">Konfirmasi</h2>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                  |
| <form role="form" id="uploadForm" enctype="multipart/form-data"             action="\$blobstoreURL" method="post">             <input type="hidden" id="idOrder" name="idOrder" value="\$order.id">             <p>                 #if(!\$device)                 <h5 style="color: #fd0202;">                     <span class="glyphicon glyphicon-exclamation-sign"></span>Perhatian:                     Konfirmasi hanya akan terkirim via email karena tidak ada device                     terdaftar.                 </h5>                 #end             </p>         </form> |                                                                                                  |



Gambar 5.51 Kode tampilan order.vm (confirm)


```

```
<div class="form-group">
    <button class="btn btn-lg btn-primary btn-block" type="submit">
        <span class="glyphicon glyphicon-envelope"></span> Kirim
    </button>
    <button onclick="window.location.href='$baseUrl/order'">
        <span class="btn btn-default btn-block">
            <span class="glyphicon glyphicon-floppy-remove"></span> Batal
        </span>
    </button>
</div>
</form>
else
-
End
```

Gambar 5.52 Lanjutan kode tampilan order.vm (*confirm*)

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    String id = request.getParameter("id");
    if (id != null) {
        FloristEndpoint api = new FloristEndpoint();
        VelocityContext context = new VelocityContext();
        BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
        String blobstoreURL = bs.createUploadUrl("/order/confirm");
        Order order = api.getOrderById(Long.valueOf(id));
        List<GCMDevices> gcmDevices = new ArrayList<GCMDevices>();
        List<GCMDevices> temp = null;
        temp = api.getGCMPersonalDevices(null, order.getSender().getId());
        if (temp.size() > 0) {
            context.put("device", true);
        } else {
            context.put("device", false);
        }
        context.put("blobstoreURL", blobstoreURL);
        context.put("baseUrl", Constant.baseUrl);
        context.put("param", "confirm");
        context.put("order", order);
        VelocityEngine ve = VelocityHelper.getVelocityEngine();
        Template template = ve.getTemplate("order.vm");
        StringWriter writer = new StringWriter();
        template.merge(context, writer);
        response.setContentType("text/html");
        response.getWriter().print(writer.toString());
    }
}
```

Gambar 5.53 Kode fungsi doGet ConfirmServlet

```

protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    String content = request.getParameter("content");
    String recipient = request.getParameter("recipient");
    String idOrder = request.getParameter("idOrder");
    String emailAddress = request.getParameter("email");
    PrintWriter io = response.getWriter();
    BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
    Map<String, BlobKey> blobs = bs.getUploadedBlobs(request);
    List<String> imagelist = new ArrayList<String>();
    List<byte[]> filelist = new ArrayList<byte[]>();
    List<String> filetypeplist = new ArrayList<String>();
    List<String> filenamelist = new ArrayList<String>();
    BlobInfoFactory bif = new BlobInfoFactory();
    for (Map.Entry<String, BlobKey> m : blobs.entrySet()) {
        BlobInfo blobInfo = bif.loadBlobInfo(m.getValue());
        if (blobInfo.getSize() != 0) {
            String contentType = blobInfo.getContentType();
            String filename = blobInfo.getFilename();
            byte[] file = bs.fetchData(m.getValue(), 0,
                BlobstoreService.MAX_BLOB_FETCH_SIZE - 1);
            imagelist.add(m.getValue().getKeyString());
            filelist.add(file);
            filetypeplist.add(contentType);
            filenamelist.add(filename);
        }
    }
    Date d = new Date();
    FloristEndpoint api = new FloristEndpoint();
    Order o = api.getOrderById(Long.valueOf(idOrder));
    Shipment s = new Shipment(imagelist, content, d, recipient, o);
    api.addShipment(s);

    try {
        MailService service = MailServiceFactory.getMailService();
        MailService.Message msg = new MailService.Message();
        msg.setSender(Constant.EMAIL_FROM);
        msg.setTo(o.getSender().getEmail());
        msg.setSubject("Konfirmasi Pemesanan Bunga MEME FLORIST");
        msg.setTextBody(content);
        List<OrderItem> orderItems = o.getItems();
        String temp = "<table style='border: 1px solid #ddd'><tr>" +
                    + "margin-right: 10px;" + "<b>Harga</b></td></tr>" +
                    for (OrderItem i : orderItems) {
                        Long hrga = i.getAmount() * i.getItem().getPrice();
                        temp += "<tr><td>" + i.getItem().getName() + "</td><td>Rp. " +
                            + i.getItem().getPrice().toString() + ",-</td><td>" +
                            + i.getAmount().toString() + "</td><td>" +
                            + hrga.toString() + "</td></tr>" +
                    }
                    temp += "<tr><td colspan='3'>Total pembayaran</b></td><td>Rp. " +
                        + o.getTotalPrice().toString() + ",-</td></tr></table>";
        String htmlBody = "<html><body><h3>Kepada pelanggan Meme Florist yang "
    ...
}

```

Gambar 5.54 Kode fungsi doPost ConfirmServlet

```
+ " <br><br>Meme Florist</p></body></html>";  
msg.setHtmlBody(htmlBody);  
Collection<MailService.Attachment> attachments = new ArrayList<MailService.Attachment>();  
for (int i = 0; i < fileList.size(); i++) {  
    byte[] file = fileList.get(i);  
    MailService.Attachment attach = new MailService.Attachment(  
        filenameList.get(i), file);  
    attachments.add(attach);  
}  
msg.setAttachments(attachments);  
service.send(msg);  
} catch (Exception e) {  
    io.println(e.getMessage());  
}  
try {  
    List<GCMDevices> gcmDevices = new ArrayList<GCMDevices>();  
    List<String> devices = new ArrayList<String>();  
    List<Long> users_id = new ArrayList<Long>();  
    List<GCMDevices> temp = null;  
    temp = api.getGCMPersonalDevices(null, o.getSender().getId());  
    gcmDevices.addAll(temp);  
    users_id.add(o.getSender().getId());  
  
    for (GCMDevices g : gcmDevices) {  
        devices.add(g.getDeviceRegId());  
    }  
  
    if (!devices.isEmpty()) {  
        for (String device : devices) {  
            io.println(device);  
        }  
        MulticastResult result = Constant.sendMessageToDevice(  
            Constant.API_KEY, devices, "Konfirmasi Pemesanan",  
            "Pesanan Anda untuk " + o.getRecipientName()  
                + " telah dikirim", "shipment", s.getId()  
                .toString());  
        io.println("API Key " + Constant.API_KEY);  
        io.println("Sukses mengirim pesan");  
        io.println("result: " + result);  
    } else  
        io.println("Gagal mengirim pesan");  
} catch (Exception e) {  
    io.println(e.getMessage());  
    io.println(e.getCause());  
}  
}  
  
response.sendRedirect("/order");  
}
```

Gambar 5.55 Lanjutan kode fungsi doPost ConfirmServlet

```
<form role="form" id="uploadForm" action="$baseUrl/sendNotification"
    method="post">
    <div class="form-group">
        <label for="category">Pilih mode</label> <select class="form-control"
            id="mode" name="mode" onchange="changeMode()">
            <option value="0">Broadcast</option>
            <option value="1">Personal</option>
        </select>
    </div>
    <div class="form-group" id="name-form" style="display: none">
        <label for="name">Nama</label> <input class="form-control" id="name"
            name="idUser" required>
    </div>
</p>
<div class="form-group">
    <label for="title">Judul pesan</label>
    <div align="right">
        <input type="text" class="form-control" placeholder="Judul pesan"
            maxlength="25" name="title" id="title" required autocomplete="off">
        <div name="title_feedback" id="title_feedback"></div>
    </div>
</div>
<div class="form-group">
    <label for="content">Isi pesan</label>
    <div align="right">
        <textarea class="form-control" placeholder="Isi pesan"
            maxlength="250" name="content" id="content" required rows="10"></textarea>
        <div name="content_feedback" id="content_feedback"></div>
    </div>
</div>
<div class="form-group">
    <button class="btn btn-lg btn-primary btn-block" type="submit">
        <span class="glyphicon glyphicon-envelope"></span> Kirim
    </button>
</div>
</form>
```

Gambar 5.56 Kode tampilan notification.vm

a. Fungsi doPost(HttpServletRequest request, HttpServletResponse response)

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang

datang, pengguna yang bersangkutan sudah dalam keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Kemudian menambahkan pemberitahuan yang telah dikirimkan untuk disimpan dengan menggunakan fungsi addNotification dari kelas FloristEndpoint. Selain itu, terdapat mekanisme pengiriman *push notification* kepada pelanggan. Setelah selesai lalu diarahkan kembali ke halaman *form* pengiriman pemberitahuan. Fungsi ini ditunjukkan oleh Gambar 5.58 dan Gambar 5.59.

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    List<User> users = api.getUsers(null);
    VelocityContext context = new VelocityContext();
    context.put("baseUrl", Constant.baseUrl);
    context.put("page", "notification");
    GsonBuilder gsonBuilder = new GsonBuilder();
    gsonBuilder.setLongSerializationPolicy(LongSerializationPolicy.STRING);
    String user_data = gsonBuilder.create().toJson(users);
    context.put("user_data", user_data);
    VelocityEngine ve = VelocityHelper.getVelocityEngine();
    Template template = ve.getTemplate("notification.vm");
    StringWriter writer = new StringWriter();
    template.merge(context, writer);
    response.setContentType("text/html");
    response.getWriter().print(writer.toString());
}
```

Gambar 5.57 Kode fungsi doGet SendNotificationServlet

5.2.10. Mengubah Profil dan Kontak Toko Meme Florist

- **Tampilan settings.vm**

Pada tampilan mengubah profil dan kontak toko Meme Florist ini terdapat beberapa kontrol, beberapa diantaranya yaitu menggunakan kontrol *Input type Text*, *Input type File*, dan *Button*. Isi dari semua atribut yang sudah tersimpan dan akan ditampilkan diisi oleh *code behind* yang ada di *SettingsServlet.java*. Kode lain dalam tampilan *settings.vm* ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.60 dan Gambar 5.61.

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    String content = request.getParameter("content");
    String title = request.getParameter("title");
    String mode = request.getParameter("mode");
    String[] idusers = request.getParameterValues("idUser[]");
    PrintWriter io = response.getWriter();

    Date d = new Date();
    Notification n = new Notification(content, d);

    try {
        List<GCMDevices> gcmDevices = new ArrayList<GCMDevices>();
        FloristEndpoint api = new FloristEndpoint();
        List<String> devices = new ArrayList<String>();

        List<User> users = new ArrayList<User>();
        List<Long> users_id = new ArrayList<Long>();

        if (mode.equals("1")) {
            for (int i = 0; i < idUsers.length; i++) {
                String idUser = idUsers[i];
                List<GCMDevice> temp = null;
                temp = api.getGCMPersonalDevices(null, Long.valueOf(idUser));
                gcmDevices.addAll(temp);
                users_id.add(Long.valueOf(idUser));
            }
        } else {
            gcmDevices = api.getGCMBroadcastDevices(null);
            users = api.getUsers(null);
            for (User u : users) {
                users_id.add(u.getId());
            }
        }
    }
```

Gambar 5.58 Kode fungsi doPost SendNotificationServlet

- **Kelas SettingsServlet.java**

Kelas ini merupakan *code behind* dari tampilan settings.vm. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan memiliki *session* sebagai administrator. Jika benar maka akan dilanjutkan ke kode berikutnya, jika tidak maka pengguna akan diarahkan menuju halaman *login*. Kemudian dengan menggunakan fungsi *getCompanyProfile* dan *getCompanyContacts* dari kelas *FloristEndpoint*, maka didapatkan profil dan kontak yang akan diubah. Setelah itu dilakukan *merging* sebagai *context* dengan *template* untuk ditampilkan oleh *Velocity Engine*. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.62.

- b. **Fungsi doPost(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP*

request berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Kemudian mengubah profil dan kontak yang telah tersimpan dengan menggunakan fungsi *addCompanyContact* dan *updateCompanyProfile* dari kelas *FloristEndpoint*. Setelah selesai lalu diarahkan kembali ke halaman menampilkan profil dan kontak yang telah disimpan. Fungsi ini ditunjukkan oleh Gambar 5.63 dan Gambar 5.64.

```
n.setUsers(users_id);
for (GCMDevices g : gcmDevices) {
    devices.add(g.getDeviceRegId());
}

if (!devices.isEmpty()) {
    for (String device : devices) {
        io.println(device);
    }
    MulticastResult result = Constant.sendMessageToDevice(Constant.API_KEY,
        devices, title, content, "notification", "");
    api.addNotification(n);
    io.println("API Key " + Constant.API_KEY);
    io.println("Sukses mengirim pesan");
    io.println("result: " + result);
} else {
    io.println("Gagal mengirim pesan");
} catch (Exception e) {
    io.println(e.getMessage());
    io.println(e.getCause());
}
response.sendRedirect("/sendNotification");
}
```

Gambar 5.59 Lanjutan kode fungsi doPost SendNotificationServlet

```
#if ( $param=="update" )
<div style="width: 103%">
<form role="form" method="post" enctype="multipart/form-data"
action="$blobstoreURL" method="post">
<div class="form-group">
    <label for="name">Username </label> <input class="form-control"
    type="text" id="username" name="username"
    value="$company.LoginUsername" required>
</div>
...
#set ($iter = 0)
<div id="contacts">
    #foreach($m in $bank)
        <div class="form-group" id="$random.get($iter)Form">
            <div class="input-group">
                <div class="input-group-btn" id="$random.get($iter)FormGroup">
                    <button type="button" id="$random.get($iter)Button"
                        class="btn btn-default dropdown-toggle" data-toggle="dropdown">
                        accountNumber <span class="caret"></span>
                    </button>
                ...
            </div>
            <div id="$random.get($iter)FormBank">
                <input type="hidden" name="$random.get($iter)accountNumber"
                value="$m.get("
                    name")|$m.get("owner")|$m.get("number")" id="$random.get($iter)Bank" />
                <div class="form-inline">
                    <div class="form-group">
                        <div class="input-group">
                            <div class="input-group-addon">Bank</div>
                            <input class="form-control" type="text"
                                onkeyup="updateBank('$random.get($iter)');"
                                id="$random.get($iter)bankName" placeholder="Nama Bank"
                                style="width: 100px" value="$m.get("name")">
                        </div>
                    </div>
                ...
            </div>
        </div>
    ...
</div>
</div>
<div class="modal fade" id="$random.get($iter)myModal" tabindex="-1"
role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
<div class="modal-dialog modal-sm">
    <div class="modal-content">
        <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal">
                <span aria-hidden="true">&times;</span><span class="sr-only">Close</span>
            </button>
            <h4 class="modal-title" id="myModalLabel">Logo bank
                "$m.get("name")"</h4>
        </div>
        <div class="modal-body">
            
        </div>
    </div>
</div>
#set ($iter = $iter+1)
#end
```

Gambar 5.60 Kode tampilan settings.vm (update)

```
#foreach($con in $contacts)
#if($con.type!="accountNumber")
<div class="form-group" id="$random.get($iter)Form">
    <div class="input-group-btn" id="$random.get($iter)FormGroup">
        <button type="button" id="$random.get($iter)Button"
            class="btn btn-default dropdown-toggle" data-toggle="dropdown">
            $con.type <span class="caret"></span>
        </button>
        ...
    ...
</div>
#set ($iter = $iter+1)
#end
#end
</div>
<button type="submit" class="btn btn-primary btn-block"
        style="margin-top: 30px; margin-bottom: 10px">Simpan</button>
<button onclick="window.location.href='$baseUrl/settings'" 
        class="btn btn-default btn-block" style="margin-bottom: 60px">Batal</button>
</form>
</div>
#else
...
#end
```

Gambar 5.61 Lanjutan kode tampilan settings.vm (*update*)

5.2.11. Melakukan Aktivasi Akun Pelanggan

- Tampilan activation.vm**

Pada tampilan melakukan aktivasi akun pelanggan tidak dibutuhkan banyak kontrol, karena kunci untuk kasus penggunaan ini terdapat pada *link* aktivasi dan *code behind* yang mengelolanya, yakni ActivationServlet.java. Kode lain dalam tampilan activation.vm ini tidak ditampilkan dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.65.

```
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    Company res = api.getCompanyProfile();
    List<CompanyContact> contacts = api.getCompanyContacts(null);
    List<Map<String, String>> map = new ArrayList<Map<String, String>>();
    for (CompanyContact c : contacts) {
        if (c.getType().equals("accountNumber")) {
            String string = c.getValue();
            String[] parts = string.split("\\|");
            Map<String, String> m = new HashMap<String, String>();
            m.put("name", parts[0]!=null?parts[0]:"");
            m.put("owner", parts[1]!=null?parts[1]:"");
            m.put("number", parts[2]!=null?parts[2]:"");
            m.put("image", parts[3]!=null?parts[3]:"");
            map.add(m);
        }
    }
    VelocityContext context = new VelocityContext();
    List<Integer> random = new ArrayList<Integer>();
    Random rand = new Random();
    for (CompanyContact c : contacts) {
        int n = rand.nextInt(1000000) + 1;
        random.add(n);
    }
    BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
    String blobstoreURL = bs.createUploadUrl("/settings");
    context.put("blobstoreURL", blobstoreURL);
    context.put("baseUrl", Constant.baseUrl);
    context.put("company", res);
    context.put("contacts", contacts);
    context.put("bank", map);
    context.put("random", random);
    context.put("page", "settings");
    String action = request.getParameter("action");
    if (action != null && action.equals("update")) {
        context.put("param", "update");
    } else {
        context.put("param", "default");
    }
    VelocityEngine ve = VelocityHelper.getVelocityEngine();
    Template template = ve.getTemplate("settings.vm");
    StringWriter writer = new StringWriter();
    template.merge(context, writer);
    response.setContentType("text/html");
    response.getWriter().print(writer.toString());
}
```

Gambar 5.62 Kode fungsi doGet SettingsServlet

```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    Constant.checkSession(response);
    FloristEndpoint api = new FloristEndpoint();
    Company res = api.getCompanyProfile();
    Company com = new Company(res.getId());
    PrintWriter out = response.getWriter();
    Enumeration<String> parameterNames = request.getParameterNames();
    api.deleteAllCompanyContact();
    BlobstoreService bs = BlobstoreServiceFactory.getBlobstoreService();
    Map<String, BlobKey> blobs = bs.getUploadedBlobs(request);
    BlobInfoFactory bif = new BlobInfoFactory();
    while (parameterNames.hasMoreElements()) {
        String paramName = parameterNames.nextElement();
        if (!paramName.equals("about") && !paramName.equals("username")
            && !paramName.equals("latitude")
            && !paramName.equals("longitude")
            && !paramName.equals("password")
            && !paramName.equals("address")) {
            if (!paramName.contains("accountNumber")
                && !paramName.contains("image")) {
                String[] paramValues = request
                    .getParameterValues(paramName);
                for (int i = 0; i < paramValues.length; i++) {
                    String paramValue = paramValues[i];
                    if (!paramValue.equals("")) {
                        CompanyContact c = new CompanyContact();
                        c.setType(paramName);
                        c.setValue(paramValue);
                        api.addCompanyContact(c);
                    }
                }
            } else if (paramName.contains("accountNumber")) {
                String id = paramName.replace("accountNumber", " ");
                out.println(id);
                boolean contain = blobs.containsKey(id + "imageUpload");
                String imageId = "";
                if (contain) {
                    BlobKey blobKey = blobs.get(id + "imageUpload");
                    BlobInfo blobInfo = bif.loadBlobInfo(blobKey);
                    if (blobInfo.getSize() != 0) {
                        imageId = blobKey.getKeyString();
                    } else{
                        imageId = " ";
                    }
                } else {
                    imageId = request.getParameter(id + "imageView");
                }
            }
        }
    }
}
```

Gambar 5.63 Kode fungsi doPost SettingsServlet

```
String paramValue = request.getParameter(paramName) + "|" + imageId;
paramName = "accountNumber";
if (!paramValue.equals("")) {
    CompanyContact c = new CompanyContact();
    c.setType(paramName);
    c.setValue(paramValue);
    api.addCompanyContact(c);
}
}

com.setAbout(new Text(request.getParameter("about")));
com.setAddress(request.getParameter("address"));
com.setLoginUsername(request.getParameter("username"));
com.setLatitude(request.getParameter("latitude"));
com.setLongitude(request.getParameter("longitude"));
com.setPassword(request.getParameter("password"));
Company updated = api.updateCompanyProfile(com);
if (updated != null) {
    response.sendRedirect("/settings");
} else {
    response.sendRedirect("/settings?action=update");
}
}
```

Gambar 5.64 Lanjutan kode fungsi doPost SettingsServlet

- **Kelas ActivationServlet.java**

Kelas ini merupakan *code behind* dari tampilan activation.vm. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain

ketika halaman pertama kali mendapatkan *request*. Di awal, fungsi ini mengecek terlebih dahulu valid tidaknya kode aktivasi yang digunakan. Kemudian dengan menggunakan fungsi setActivated dan updateUser dari kelas FloristEndpoint, maka akun pelanggan telah selesai diaktivasi. Setelah itu menampilkan *template* oleh *Velocity Engine*. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.66.

```
<div class="cover-container">
  <div class="inner cover">
    
    <h1 class="cover-heading">Selamat, konfirmasi pendaftaran berhasil!</h1>
    <p class="lead">Silahkan login ke akun Anda dengan menggunakan aplikasi mobile Meme Florist</p>
    <p class="lead">
      <a href="http://www.memeflorist.com" class="btn btn-lg btn-default">Kunjungi kami</a>
    </p>
  </div>

  <div class="mastfoot">
    <div class="inner">
      <p>
        <a href="http://www.memeflorist.com">Meme Florist</a>
      </p>
    </div>
  </div>
</div>
```

Gambar 5.65 Kode tampilan activation.vm

5.2.12. Mengubah *Password* Akun Pelanggan

- **Tampilan reset.vm**

Pada tampilan mengubah *password* akun pelanggan tidak dibutuhkan banyak kontrol, karena kunci untuk kasus penggunaan ini terdapat pada *link* permintaan *reset password* dan *code behind* yang mengelolanya, yakni ResetPasswordServlet.java. Kode lain dalam tampilan reset.vm ini tidak ditampilkan

dalam bagian laporan ini. Kode sumber tampilan ditunjukkan oleh Gambar 5.67.

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String id = request.getParameter("confirm_id");
    if (id != null) {
        FloristEndpoint api = new FloristEndpoint();
        User u = api.getUserById(Long.valueOf(id));
        if (u != null) {
            u.setActivated(true);
            api.updateUser(u);
            VelocityContext context = new VelocityContext();
            VelocityEngine ve = VelocityHelper.getVelocityEngine();
            Template template = ve.getTemplate("activation.vm");
            StringWriter writer = new StringWriter();
            template.merge(context, writer);
            response.setContentType("text/html");
            response.getWriter().print(writer.toString());
        } else {
            Constant.showResult(response, "Maaf, aktivasi akun Anda gagal",
                "Kode aktivasi Anda tidak ditemukan");
        }
    } else {
        Constant.notFound(response);
    }
}
```

Gambar 5.66 Kode fungsi doGet ActivationServlet

```
<form action="resetPassword" method="post" id="formReset"
onsubmit="return submitForm();">

<h2 class="form-signin-heading">Reset password</h2>
<input type="hidden" name="reset_id" value="$reset_id"> <input
    type="email" class="form-control" placeholder="Email" name="email"
    required autofocus> <input type="password"
    class="form-control" placeholder="Password" id="password"
    name="password" required> <input type="password"
    class="form-control" placeholder="Confirm Password"
    name="confirm_password" id="confirm_password"
    style="margin-top: -10px" required>
<div id="notif"></div>
<br />
<button class="btn btn-lg btn-primary btn-block"
    style="margin-left: 0px">Submit</button>
</form>
```

Gambar 5.67 Kode tampilan reset.vm

- **Kelas ResetPasswordServlet.java**

Kelas ini merupakan *code behind* dari tampilan reset.vm. Fungsi yang terlibat di dalamnya terdiri dari beberapa potongan bagian kode yang antara lain sebagai berikut:

- a. **Fungsi doGet(HttpServletRequest request, HttpServletResponse response)**

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *get*, atau dengan kata lain ketika halaman pertama kali mendapatkan *request*. Di awal, fungsi ini mengecek terlebih dahulu valid tidaknya kode permintaan *reset password* yang digunakan. Setelah itu menampilkan *template* oleh *Velocity Engine*. Kode sumber untuk fungsi ini ditunjukkan oleh Gambar 5.68.

```
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
    String id = request.getParameter("reset_id");
    if (id != null) {
        FloristEndpoint api = new FloristEndpoint();
        Password p = api.getPassword(Long.valueOf(id));
        if (p != null) {
            VelocityContext context = new VelocityContext();
            VelocityEngine ve = VelocityHelper.getVelocityEngine();
            Template template = ve.getTemplate("reset.vm");
            StringWriter writer = new StringWriter();
            context.put("reset_id", id);
            template.merge(context, writer);
            response.setContentType("text/html");
            response.getWriter().print(writer.toString());
        } else {
            Constant.showResult(response,
                "Maaf, reset password Anda ditolak",
                "Permintaan lupa password Anda tidak ditemukan");
        }
    } else {
        Constant.notFound(response);
    }
}
```

Gambar 5.68 Kode fungsi doGet ResetPasswordServlet

b. Fungsi doPost(HttpServletRequest request, HttpServletResponse response)

Fungsi ini digunakan untuk mengeksekusi ketika ada permintaan dari klien dengan jenis *HTTP request* berupa *post*, atau dengan kata lain biasanya setelah halaman hasil dari *request* tipe *get* melakukan *submit form* dengan metode *post*. Di dalam fungsi ini diawali dengan mekanisme pengecekan apakah ketika ada *request* yang datang, pengguna yang bersangkutan sudah dalam keadaan sudah *login*. Jika sudah maka pengguna akan langsung diarahkan ke halaman beranda, selain itu maka akan dilanjutkan ke kode berikutnya. Dengan *value* yang dikirimkan oleh *request* berupa *email* dan *password* yang baru, kemudian mengganti *password* lama dengan yang baru dengan menggunakan fungsi *updatePassword* dari kelas *FloristEndpoint*. Setelah selesai lalu diarahkan kembali ke halaman menampilkan notifikasi *reset password* telah sukses. Fungsi ini ditunjukkan oleh Gambar 5.69.

5.2.13. API (*Application Programming Interface*) Florist Endpoint

API Florist Endpoint adalah suatu kelas yang nantinya akan dibangkitkan menjadi *Endpoint Class*. Hal ini penting karena *Android Client* akan mengakses data melalui Google Cloud Endpoint, sehingga dibutuhkan suatu *interface* khusus seperti API ini. Di dalam kelas API ini terdapat beberapa fungsi yang memiliki kegunaan berbeda-beda, antara lain sebagai berikut:

- **Fungsi getCatalogs**

Fungsi ini untuk melakukan *query* semua data katalog yang telah disimpan. Kode fungsi ini ditunjukkan oleh Gambar 5.70.

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
    String email = request.getParameter("email");
    String password = request.getParameter("password");
    String id = request.getParameter("reset_id");
    if (id != null) {
        FloristEndpoint api = new FloristEndpoint();
        User u = api.loginGplus(email);
        Password p = api.getPassword(Long.valueOf(id));
        if (p != null && u != null) {
            p.setUser(u);
            api.updatePassword(p, password);

            VelocityContext context = new VelocityContext();
            VelocityEngine ve = VelocityHelper.getVelocityEngine();
            Template template = ve.getTemplate("resetSuccess.vm");
            StringWriter writer = new StringWriter();
            template.merge(context, writer);
            response.setContentType("text/html");
            response.getWriter().print(writer.toString());
        } else {
            Constant.showResult(response,
                                "Maaf, reset password Anda gagal",
                                "Email atau permintaan lupa password Anda tidak ditemukan");
        }
    } else {
        Constant.notFound(response);
    }
}
```

Gambar 5.69 Kode fungsi doPost ResetPasswordServlet

- **Fungsi getCategories**

Fungsi ini untuk melakukan *query* semua data kategori yang telah disimpan. Kode fungsi ini ditunjukkan oleh Gambar 5.71.

```
@SuppressWarnings({ "unchecked" })
@ApiMethod(name = "getCatalogs")
public List<Catalog> getCatalogs(@Nullable @Named("limit") Integer limit) {

    PersistenceManager mgr = PMF.get().getPersistenceManager();
    List<Catalog> catalogs = null;

    try {
        Query query = mgr.newQuery(Catalog.class);
        if (limit != null) {
            query.setRange(0, limit);
        }
        catalogs = (List<Catalog>) query.execute();
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return catalogs;
}
```

Gambar 5.70 Kode fungsi getCatalogs

```
@SuppressWarnings({ "unchecked" })
@ApiMethod(name = "getCategories")
public List<Category> getCategories(@Nullable @Named("limit") Integer limit) {

    PersistenceManager mgr = getPersistenceManager();
    List<Category> categories = null;

    try {
        Query query = mgr.newQuery(Category.class);
        if (limit != null) {
            query.setRange(0, limit);
        }
        categories = (List<Category>) query.execute();
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return categories;
}
```

Gambar 5.71 Kode fungsi getCategories

- **Fungsi getGCMBroadcastDevices**

Fungsi ini untuk melakukan *query* semua data *device* pelanggan yang diperbolehkan untuk menerima *push*

notification secara *broadcast*. Kode fungsi ini ditunjukkan oleh Gambar 5.74.

- **Fungsi getGCMPersonalDevices**

Fungsi ini untuk melakukan *query* semua data *device* pelanggan yang diperbolehkan untuk menerima *push notification* secara *personal*. Kode fungsi ini ditunjukkan oleh Gambar 5.75.

```
@SuppressWarnings({"unchecked"})
@apiMethod(name = "getCompanyContacts")
public List<CompanyContact> getCompanyContacts(
    @Nullable @Named("limit") Integer limit) {

    PersistenceManager mgr = getPersistenceManager();
    List<CompanyContact> companyContact = null;

    try {
        Query query = mgr.newQuery(CompanyContact.class);
        query.setOrdering("type asc");
        if (limit != null) {
            query.setRange(0, limit);
        }
        companyContact = (List<CompanyContact>) query.execute();
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return companyContact;
}
```

Gambar 5.72 Kode fungsi getCompanyContacts

- **Fungsi getOrders**

Fungsi ini untuk melakukan *query* semua data pemesanan pelanggan yang telah disimpan. Kode fungsi ini ditunjukkan oleh Gambar 5.76.

- **Fungsi getCompanyProfile**

Fungsi ini untuk melakukan *query* data profil toko Meme Florist yang telah disimpan. Kode fungsi ini ditunjukkan oleh Gambar 5.77.

```
public List<User> getUsers(@Nullable @Named("limit") Integer limit) {  
    PersistenceManager mgr = getPersistenceManager();  
    List<User> users = null;  
  
    try {  
        Query query = mgr.newQuery(User.class);  
        if (limit != null) {  
            query.setRange(0, limit);  
        }  
        users = (List<User>).query.execute();  
    } catch (Exception ex) {  
        return null;  
    } finally {  
        mgr.close();  
    }  
  
    return users;  
}
```

Gambar 5.73 Kode fungsi getUsers

```
public List<GCMDevices> getGCMBroadcastDevices(  
    @Nullable @Named("limit") Integer limit) {  
  
    PersistenceManager mgr = getPersistenceManager();  
    List<GCMDevices> devices = null;  
  
    try {  
        Query query = mgr.newQuery(GCMDevices.class);  
        query.setFilter("acceptBroadcastNotification == acceptBroadcastNotificationParam");  
        query.declareParameters("Boolean acceptBroadcastNotificationParam");  
        if (limit != null) {  
            query.setRange(0, limit);  
        }  
  
        devices = (List<GCMDevices>).query.execute(true);  
    } catch (Exception ex) {  
        return null;  
    } finally {  
        mgr.close();  
    }  
  
    return devices;  
}
```

Gambar 5.74 Kode fungsi getGCMBroadcastDevices

```
@SuppressWarnings({ "unchecked" })
@ApiMethod(name = "getGCMPersonalDevices")
public List<GCMDevices> getGCMPersonalDevices(
    @Nullable @Named("limit") Integer limit,
    @Named("idUser") Long idUser) {

    PersistenceManager mgr = getPersistenceManager();
    List<GCMDevices> devices = null;

    try {
        Query query = mgr.newQuery(GCMDevices.class);
        User u = new User(idUser);
        query.setFilter("acceptPersonalNotification == acceptPersonalNotificationParam "
            + " && user == userParam");
        query.declareParameters("Boolean acceptPersonalNotificationParam, User userParam");
        if (limit != null) {
            query.setRange(0, limit);
        }

        devices = (List<GCMDevices>) query.execute(true, u);
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return devices;
}
```

Gambar 5.75 Kode fungsi getGCMPersonalDevices

```
@SuppressWarnings({ "unchecked" })
@ApiMethod(name = "getOrders")
public List<Order> getOrders(@Nullable @Named("limit") Integer limit) {

    PersistenceManager mgr = getPersistenceManager();
    List<Order> orders = null;

    try {
        Query query = mgr.newQuery(Order.class);
        if (limit != null) {
            query.setRange(0, limit);
        }
        orders = (List<Order>) query.execute();
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return orders;
}
```

Gambar 5.76 Kode fungsi getOrders

- **Fungsi loginCompany**

Fungsi ini untuk melakukan *login* oleh administrator di aplikasi *web*. Kode fungsi ini ditunjukkan oleh Gambar 5.78.

```
@ApiMethod(name = "getCompanyProfile")
public Company getCompanyProfile() {

    PersistenceManager mgr = getPersistenceManager();
    List<Company> result = null;

    try {
        Query query = mgr.newQuery(Company.class);
        query.setRange(0, 1);

        result = (List<Company>) query.execute();
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return result.get(0);
}
```

Gambar 5.77 Kode fungsi getCompanyProfile

- **Fungsi historyOrders**

Fungsi ini untuk melakukan *query* daftar pemesanan yang telah dilakukan oleh seorang pelanggan. Kode fungsi ini ditunjukkan oleh Gambar 5.80.

- **Fungsi historyNotifications**

Fungsi ini untuk melakukan *query* daftar pemberitahuan atau *push notification* yang telah diterima oleh seorang pelanggan. Kode fungsi ini ditunjukkan oleh Gambar 5.81.

```
public Company loginCompany(@Named("username") String username,
                           @Named("password") String password) {

    PersistenceManager mgr = getPersistenceManager();
    List<Company> result = null;

    try {
        Query query = mgr.newQuery(Company.class);
        query.setFilter("loginUsername == usernameParam && password == passwordParam");
        query.declareParameters("String usernameParam, String passwordParam");
        query.setRange(0, 1);
        result = (List<Company>) query.execute(username, password);
    } catch (Exception e) {
        return null;
    } finally {
        mgr.close();
    }

    if (result.size() > 0)
        return result.get(0);
    else
        return null;
}
```

Gambar 5.78 Kode fungsi loginCompany

```
@ApiMethod(name = "getCatalogsByCategory")
public List<Catalog> getCatalogsByCategory(
    @Named("idCategory") Long idCategory) {

    PersistenceManager mgr = getPersistenceManager();
    List<Catalog> catalogs = null;

    try {
        Query q = mgr.newQuery(Catalog.class);
        q.setFilter("category == categoryParam");
        q.declareParameters("Category categoryParam");
        Category c = new Category(idCategory);
        catalogs = (List<Catalog>) q.execute(c);

    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return catalogs;
}
```

Gambar 5.79 Kode fungsi getCatalogsByCategory

- **Fungsi historyShipments**

Fungsi ini untuk melakukan *query* daftar pengiriman atau konfirmasi dari pemesanan yang telah dilakukan oleh seorang pelanggan. Kode fungsi ini ditunjukkan oleh Gambar 5.82.

```
@ApiMethod(name = "historyOrders")
public List<Order> historyOrders(@Named("idUser") Long idUser) {
    List<Order> orders = null;
    PersistenceManager mgr = getPersistenceManager();

    try {
        Query q = mgr.newQuery(Order.class);
        q.setFilter("sender == senderParam");
        q.declareParameters("User senderParam");
        User u = new User(idUser);
        orders = (List<Order>) q.execute(u);

    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return orders;
}
```

Gambar 5.80 Kode fungsi historyOrders

```
@ApiMethod(name = "historyNotifications")
public List<Notification> historyNotifications(@Named("idUser") Long idUser) {
    List<Notification> notifications = null;
    PersistenceManager mgr = getPersistenceManager();

    try {
        Query q = mgr.newQuery(Notification.class);
        q.setFilter(":usersParam.contains(users)");
        notifications = (List<Notification>) q.execute(Arrays
                .asList(idUser));
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return notifications;
}
```

Gambar 5.81 Kode fungsi historyNotifications

```
@ApiMethod(name = "historyShipments")
public List<Shipment> historyShipments(@Named("idUser") Long idUser) {
    List<Shipment> shipments = new ArrayList<Shipment>();
    List<Order> orders = null;
    PersistenceManager mgr = getPersistenceManager();

    try {
        Query q1 = mgr.newQuery(Order.class);
        q1.setFilter("sender == senderParam");
        q1.declareParameters("User senderParam");
        User u = new User(idUser);
        orders = (List<Order>) q1.execute(u);

        for (Order o : orders) {
            List<Shipment> temp = null;
            Query q2 = mgr.newQuery(Shipment.class);
            q2.setFilter("order == orderParam");
            q2.declareParameters("Order orderParam");
            temp = (List<Shipment>) q2.execute(o);
            shipments.addAll(temp);
        }
    } catch (Exception ex) {
        return null;
    } finally {
        mgr.close();
    }

    return shipments;
}
```

Gambar 5.82 Kode fungsi historyShipments

- **Fungsi getCatalogById**

Fungsi ini untuk melakukan *query* katalog tertentu berdasarkan ID katalog. Kode fungsi ini ditunjukkan oleh Gambar 5.83.

- **Fungsi getShipmentById**

Fungsi ini untuk melakukan *query* pengiriman atau konfirmasi dari pemesanan yang telah dilakukan oleh seorang pelanggan berdasarkan ID pengiriman. Kode fungsi ini ditunjukkan oleh Gambar 5.84.

```
@ApiMethod(name = "getCatalogById")
public Catalog getCatalogById(@Named("idCatalog") Long idCatalog) {

    PersistenceManager mgr = getPersistenceManager();
    Catalog catalog = null;

    try {
        catalog = mgr.getObjectById(Catalog.class, idCatalog);
    } catch (javax.jdo.JDOObjectNotFoundException ex) {
        return null;
    } finally {
        mgr.close();
    }

    return catalog;
}
```

Gambar 5.83 Kode fungsi getCatalogById

```
@ApiMethod(name = "getShipmentById")
public Shipment getShipmentById(@Named("idCatalog") Long idShipment) {

    PersistenceManager mgr = getPersistenceManager();
    Shipment shipment = null;

    try {
        shipment = mgr.getObjectById(Shipment.class, idShipment);
    } catch (javax.jdo.JDOObjectNotFoundException ex) {
        return null;
    } finally {
        mgr.close();
    }

    return shipment;
}
```

Gambar 5.84 Kode fungsi getShipmentById

- **Fungsi getUserById**

Fungsi ini untuk melakukan *query* data pelanggan berdasarkan ID pelanggan. Kode fungsi ini ditunjukkan oleh Gambar 5.85.

```
@ApiMethod(name = "getUserById")
public User getUserById(@Named("idUser") Long idUser) {

    PersistenceManager mgr = getPersistenceManager();
    User user = null;

    try {
        user = mgr.getObjectById(User.class, idUser);
    } catch (javax.jdo.JDOObjectNotFoundException ex) {
        return null;
    } finally {
        mgr.close();
    }

    return user;
}
```

Gambar 5.85 Kode fungsi getUserId

- **Fungsi getCategoryById**

Fungsi ini untuk melakukan *query* kategori berdasarkan ID kategori. Kode fungsi ini ditunjukkan oleh Gambar 5.86.

```
@ApiMethod(name = "getCategoryById")
public Category getCategoryById(@Named("id") Long id) {

    PersistenceManager mgr = getPersistenceManager();
    Category category = null;

    try {
        category = mgr.getObjectById(Category.class, id);
    } catch (javax.jdo.JDOObjectNotFoundException ex) {
        return null;
    } finally {
        mgr.close();
    }

    return category;
}
```

Gambar 5.86 Kode fungsi getCategoryById

- **Fungsi getOrderById**

Fungsi ini untuk melakukan *query* pemesanan pelanggan berdasarkan ID pemesanannya. Kode fungsi ini ditunjukkan oleh Gambar 5.87.

```
@ApiMethod(name = "getOrderById")
public Order getOrderById(@Named("id") Long id) {

    PersistenceManager mgr = getPersistenceManager();
    Order order = null;

    try {
        order = mgr.getObjectById(Order.class, id);
    } catch (javax.jdo.JDOObjectNotFoundException ex) {
        return null;
    } finally {
        mgr.close();
    }

    return order;
}
```

Gambar 5.87 Kode fungsi getOrderById

- **Fungsi addCatalog**

Fungsi ini untuk menambahkan katalog. Kode fungsi ini ditunjukkan oleh Gambar 5.88.

```
public void addCatalog(Catalog catalog) {
    PersistenceManager mgr = getPersistenceManager();
    try {
        if (catalog.getId() != null) {
            if (containsCatalog(catalog)) {
                throw new EntityExistsException("Object already exists");
            }
        }
        Category c = mgr.getObjectById(Category.class, catalog
            .getCategory().getId());
        catalog.setCategory(c);
        mgr.makePersistent(catalog);
    } finally {
        mgr.close();
    }
}
```

Gambar 5.88 Kode fungsi addCatalog

- **Fungsi addCompanyProfile**

Fungsi ini untuk menambahkan profil toko Meme Florist. Kode fungsi ini ditunjukkan oleh Gambar 5.89.

```
public void addCompanyProfile(Company company) {  
    PersistenceManager mgr = getPersistenceManager();  
    try {  
        if (company.getId() != null) {  
            if (containsCompany(company)) {  
                throw new EntityExistsException("Object already exists");  
            }  
        }  
        mgr.makePersistent(company);  
    } finally {  
        mgr.close();  
    }  
}
```

Gambar 5.89 Kode fungsi addCompanyProfile

- **Fungsi addCompanyContact**

Fungsi ini untuk menambahkan kontak toko Meme Florist. Kode fungsi ini ditunjukkan oleh Gambar 5.90.

```
public void addCompanyContact(CompanyContact companyContact) {  
    PersistenceManager mgr = getPersistenceManager();  
    try {  
        if (companyContact.getId() != null) {  
            if (containsCompanyContact(companyContact)) {  
                throw new EntityExistsException("Object already exists");  
            }  
        }  
        mgr.makePersistent(companyContact);  
    } finally {  
        mgr.close();  
    }  
}
```

Gambar 5.90 Kode fungsi addCompanyContact

- **Fungsi addOrder**

Fungsi ini untuk menambahkan pemesanan pelanggan.

Kode fungsi ini ditunjukkan oleh Gambar 5.91.

```
@ApiMethod(name = "addOrder")
public void addOrder(Order order) {
    PersistenceManager mgr = getPersistenceManager();
    try {
        if (order.getId() != null) {
            if (containsOrder(order)) {
                throw new EntityExistsException("Object already exists");
            }
        }
        List<OrderItem> orderItems = new ArrayList<OrderItem>();
        List<OrderItem> items = order.getItems();
        for (OrderItem i : items) {
            Catalog newC = mgr.getObjectById(Catalog.class, i.getItem()
                .getId());
            i.setItem(newC);
            mgr.makePersistent(i);
            orderItems.add(i);
        }
        order.setStatus(false);
        order.setItems(orderItems);
        User newU = mgr
            .getObjectById(User.class, order.getSender().getId());
        order.setSender(newU);
        mgr.makePersistent(order);
    } finally {
        mgr.close();
    }
}
```

Gambar 5.91 Kode fungsi addOrder

- **Fungsi addNotification**

Fungsi ini untuk menambahkan *push notification* untuk disimpan. Kode fungsi ini ditunjukkan oleh Gambar 5.92.

- **Fungsi addShipment**

Fungsi ini untuk menambahkan pengiriman bahwa pemesanan telah selesai diproses atau dikirim ke tujuan. Kode fungsi ini ditunjukkan oleh Gambar 5.93.

```
public void addNotification(Notification notification) {  
    PersistenceManager mgr = getPersistenceManager();  
    try {  
        if (notification.getId() != null) {  
            if (containsNotification(notification)) {  
                throw new EntityExistsException("Object already exists");  
            }  
        }  
        mgr.makePersistent(notification);  
    } finally {  
        mgr.close();  
    }  
}
```

Gambar 5.92 Kode fungsi addNotification

```
public void addShipment(Shipment shipment) {  
    PersistenceManager mgr = getPersistenceManager();  
    Order o = null;  
    Order temp1, temp2 = null;  
    try {  
        if (shipment.getId() != null) {  
            if (containsShipment(shipment.order.getId()) != null) {  
                throw new EntityExistsException("Object already exists");  
            }  
        }  
        o = mgr.getObjectById(Order.class, shipment.order.getId());  
        o.setStatus(true);  
        temp1 = updateOrder(o, mgr);  
        shipment.setOrder(temp1);  
        mgr.makePersistent(shipment);  
    } finally {  
        mgr.close();  
    }  
}
```

Gambar 5.93 Kode fungsi addShipment

- **Fungsi registerUser**

Fungsi ini untuk mendaftarkan pelanggan baru. Kode fungsi ini ditunjukkan oleh Gambar 5.94.

- **Fungsi registerDevice**

Fungsi ini untuk mendaftarkan *device* pelanggan agar dapat menerima *push notification*. Kode fungsi ini ditunjukkan oleh Gambar 5.95.

```
@ApiMethod(name = "registerUser")
public User registerUser(User user, @Named("password") String password) {
    PersistenceManager mgr = getPersistenceManager();
    PasswordEncryptionService passService = new PasswordEncryptionService();
    Password pass = null;
    if (!password.equals(" ")) {
        try {
            byte[] salt = passService.generateSalt();
            byte[] getEncrypted = passService.getEncryptedPassword(
                password, salt);
            String saltString = Base64.encodeBase64String(salt);
            String getEncryptedString = Base64
                .encodeBase64String(getEncrypted);
            pass = new Password(getEncryptedString, saltString, user);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
    }
    try {
        if (containsUser(user.getEmail())) {
            return null;
        }
        if (!password.equals(" ")) {
            mgr.makePersistent(pass);
        }
        user.setActivated(false);
        mgr.makePersistent(user);
        sendEmailRegistration(user);
        return user;
    } finally {
        mgr.close();
    }
}
```

Gambar 5.94 Kode fungsi registerUser

- **Fungsi loginGplus**

Fungsi ini untuk *login* di *device* pelanggan dengan menggunakan akun Google Plus. Kode fungsi ini ditunjukkan oleh Gambar 5.96.

- **Fungsi loginDefault**

Fungsi ini untuk *login* di *device* pelanggan dengan menggunakan akun *default* Meme Florist. Kode fungsi ini ditunjukkan oleh Gambar 5.97.

```
@ApiMethod(name = "registerDevice")
public GCMDevices registerDevice(GCMDevices device) {
    PersistenceManager mgr = getPersistenceManager();
    GCMDevices getDevice = null;
    try {
        getDevice = containsGCMDevices(device.getDeviceRegId());
        if (getDevice != null) {
            device.setId(getDevice.getId());
            GCMDevices updatedDevice = null;
            updatedDevice = updateDevice(device);
            return updatedDevice;
        }
        mgr.makePersistent(device);
        return device;
    } finally {
        mgr.close();
    }
}
```

Gambar 5.95 Kode fungsi registerDevice

```
@ApiMethod(name = "loginGplus")
public User loginGplus(@Named("email") String email) {

    List<User> res = null;
    PersistenceManager mgr = getPersistenceManager();
    boolean match = true;
    User u = null;
    try {
        Query q = mgr.newQuery(User.class);
        q.setFilter("email == emailParam && activated == activatedParam");
        q.declareParameters("String emailParam, Boolean activatedParam");
        q.setRange(0, 1);
        res = (List<User>).q.execute(email, true);
        if (res.size() > 0) {
            return res.get(0);
        } else {
            return null;
        }
    } finally {
        mgr.close();
    }
}
```

Gambar 5.96 Kode fungsi loginGplus

```
@ApiMethod(name = "loginDefault")
public User loginDefault(@Named("email") String email,
    @Named("password") String password) {
    List<User> res = null;
    List<Password> pass = null;
    PersistenceManager mgr = getPersistenceManager();
    boolean match = true;
    User u = null;
    Password p = null;
    PasswordEncryptionService passService = new PasswordEncryptionService();
    try {
        Query q = mgr.newQuery(User.class);
        q.setFilter("email == emailParam && activated == activatedParam");
        q.declareParameters("String emailParam, Boolean activatedParam");
        q.setRange(0, 1);
        res = (List<User>).q.execute(email, true);
        if (res.size() > 0) {
            u = res.get(0);
            q = mgr.newQuery(Password.class);
            q.setFilter("user == userParam");
            q.declareParameters("User userParam");
            q.setRange(0, 1);
            pass = (List<Password>).q.execute(u);
            if (res.size() > 0) {
                p = pass.get(0);
                byte[] passByte = Base64.decodeBase64(p.getPassword()
                    .getBytes());
                byte[] saltByte = Base64.decodeBase64(p.getSalt()
                    .getBytes());
                match = passService.authenticate(password, passByte,
                    saltByte);
                if (match) {
                    return u;
                } else {
                    return null;
                }
            } else {
                return null;
            }
        } else {
            return null;
        }
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    } catch (InvalidKeySpecException e) {
        e.printStackTrace();
        return null;
    } finally {
        mgr.close();
    }
}
```

Gambar 5.97 Kode fungsi loginDefault

- **Fungsi forgotPassword**

Fungsi ini untuk mengirimkan permintaan *reset password* yang nantinya akan dikirim ke *email*

pelanggan yang bersangkutan. Kode fungsi ini ditunjukkan oleh Gambar 5.98.

```
@ApiMethod(name = "forgotPassword")
public void forgotPassword(@Named("email") String email) {
    List<User> res = null;
    List<Password> pass = null;
    PersistenceManager mgr = getPersistenceManager();
    boolean match = true;
    User u = null;
    Password p = null;
    try {
        Query q = mgr.newQuery(User.class);
        q.setFilter("email == emailParam && activated == activatedParam");
        q.declareParameters("String emailParam, Boolean activatedParam");
        q.setRange(0, 1);
        res = (List<User>) q.execute(email, true);
        if (res.size() > 0) {
            u = res.get(0);
            q = mgr.newQuery(Password.class);
            q.setFilter("user == userParam");
            q.declareParameters("User userParam");
            q.setRange(0, 1);
            pass = (List<Password>) q.execute(u);
            if (pass.size() > 0) {
                p = pass.get(0);
                sendResetPassword(email, p);
            }
        }
    } catch (Exception e) {
    } finally {
        mgr.close();
    }
}
```

Gambar 5.98 Kode fungsi forgotPassword

- **Fungsi updateCatalog**

Fungsi ini untuk mengubah data katalog. Kode fungsi ini ditunjukkan oleh Gambar 5.99.

- **Fungsi updateDevice**

Fungsi ini untuk mengubah data *device*. Kode fungsi ini ditunjukkan oleh Gambar 5.100.

- **Fungsi updateCompanyProfile**

Fungsi ini untuk mengubah data profil toko Meme Florist. Kode fungsi ini ditunjukkan oleh Gambar 5.101.

```
public Catalog updateCatalog(Catalog catalog) {  
    PersistenceManager mgr = getPersistenceManager();  
  
    try {  
        if (!containsCatalog(catalog)) {  
            throw new EntityNotFoundException("Object does not exist");  
        }  
        mgr.makePersistent(catalog);  
    } finally {  
        mgr.close();  
    }  
    return catalog;  
}
```

Gambar 5.99 Kode fungsi updateCatalog

```
public GCMDevices updateDevice(GCMDevices device) {  
    PersistenceManager mgr = getPersistenceManager();  
  
    try {  
        if (containsGCMDevices(device.getDeviceRegId()) == null) {  
            throw new EntityNotFoundException("Object does not exist");  
        }  
        if (device.getUser() != null) {  
            User newU = mgr.getObjectById(User.class, device.getUser()  
                .getId());  
            device.setUser(newU);  
        }  
        mgr.makePersistent(device);  
    } finally {  
        mgr.close();  
    }  
    return device;  
}
```

Gambar 5.100 Kode fungsi updateDevice

```
public Company updateCompanyProfile(Company company) {  
    PersistenceManager mgr = getPersistenceManager();  
    try {  
        if (!containsCompany(company)) {  
            throw new EntityNotFoundException("Object does not exist");  
        }  
        mgr.makePersistent(company);  
    } finally {  
        mgr.close();  
    }  
    return company;  
}
```

Gambar 5.101 Kode fungsi updateCompanyProfile

- **Fungsi updateUser**

Fungsi ini untuk mengubah data pelanggan. Kode fungsi ini ditunjukkan oleh Gambar 5.102.

```
@ApiMethod(name = "updateUser")
public User updateUser(User user) {
    PersistenceManager mgr = getPersistenceManager();
    try {
        if (!containsUser(user)) {
            throw new EntityNotFoundException("Object does not exist");
        }
        mgr.makePersistent(user);
    } finally {
        mgr.close();
    }
    return user;
}
```

Gambar 5.102 Kode fungsi updateUser

- **Fungsi updateOrder**

Fungsi ini untuk mengubah data pemesanan seperti mengubah status pemesanan setelah dikonfirmasi. Kode fungsi ini ditunjukkan oleh Gambar 5.103.

```
private Order updateOrder(Order order, PersistenceManager mgr) {
    try {
        if (!containsOrder(order)) {
            throw new EntityNotFoundException("Object does not exist");
        }
        mgr.makePersistent(order);
    } finally {
    }
    return order;
}
```

Gambar 5.103 Kode fungsi updateOrder

- **Fungsi updatePassword**

Fungsi ini untuk mengubah *password* pelanggan melalui permintaan *reset password*. Kode fungsi ini ditunjukkan oleh Gambar 5.104.

- **Fungsi removeCatalogById**

Fungsi ini untuk menghapus katalog yang dipilih. Kode fungsi ini ditunjukkan oleh Gambar 5.105.

```
public Password updatePassword(Password password,
    @Named("new_password") String new_pass) {
    PersistenceManager mgr = getPersistenceManager();
    PasswordEncryptionService passService = new PasswordEncryptionService();
    if (!new_pass.equals(" ")) {
        try {
            byte[] salt = passService.generateSalt();
            byte[] getEncrypted = passService.getEncryptedPassword(
                new_pass, salt);
            String saltString = Base64.encodeBase64String(salt);
            String getEncryptedString = Base64
                .encodeBase64String(getEncrypted);
            password.setPassword(getEncryptedString);
            password.setSalt(saltString);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
    }
    try {
        if (!containsPassword(password)) {
            throw new EntityNotFoundException("Object does not exist");
        }
        mgr.makePersistent(password);
    } finally {
        mgr.close();
    }
    return password;
}
```

Gambar 5.104 Kode fungsi updatePassword

```
public void removeCatalogById(@Named("id") Long id) {
    PersistenceManager mgr = getPersistenceManager();
    try {
        Catalog catalog = mgr.getObjectById(Catalog.class, id);
        mgr.deletePersistent(catalog);
    } finally {
        mgr.close();
    }
}
```

Gambar 5.105 Kode fungsi removeCatalogById

BAB VI

UJI COBA DAN EVALUASI

Pada bab ini, akan dijelaskan mengenai proses uji coba yang dilakukan untuk setiap fungsionalitas dari aplikasi Meme Florist dengan Google App Engine berbasis perangkat *Mobile Android* untuk pemesanan bunga online. Selain itu, akan diuraikan juga hasil evaluasi dari uji coba tersebut.

6.1 Aplikasi Perangkat *Mobile* Meme Florist

Pada subbab ini, akan dijelaskan mengenai proses ujicoba pada perangkat *mobile* untuk setiap fungsionalitas Meme Florist.

6.1.1 Lingkungan Uji Coba

Lingkungan Uji Coba yang digunakan dalam proses pengembangan aplikasi Meme Florist dengan Google App Engine berbasis perangkat *Mobile Android* untuk pemesanan bunga online ini adalah Microsoft Windows 7, Java Runtime Environment 7, dan perangkat pengembang yang digunakan adalah Eclipse Kepler 4.3.0, Java Development Kit 1.7.0_60. Lingkungan uji coba lain yang juga digunakan adalah beberapa perangkat *mobile* yang memiliki sistem operasi Android minimum versi 4.0.

6.1.2 Menampilkan Hasil Uji Coba

a. Menampilkan halaman *Splashscreen*

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan *splashscreen* di awal aplikasi dijalankan yang ditunjukkan oleh Gambar 6.1.

b. Menampilkan halaman *Landing Menu* pengguna yang tidak melakukan *login*

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman *landing menu*. Pada gambar Gambar 6.2 dapat dilihat bahwa pengguna mendapatkan 3 menu utama yang dapat

digunakan sebelum pengguna melakukan *login*. Meenu tersebut adalah katalog, tentang kami, dan hubungi kami. Serta terdapat *icon login* yang berada di ujung atas kanan.

c. Menampilkan halaman kategori bunga dan detail bunga

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman kategori bunga dan detail bunga yang ditunjukkan dengan Gambar 6.3 dan Gambar 6.4.

Pada gambar Gambar 6.4 dapat dilihat bahwa seluruh kategori bunga ditampilkan dan terdapat *icon login* bagi pengguna.

Pada Gambar 6.3 dapat dilihat bahwa detail bunga ditampilkan dengan gambar bunga, kode bunga, ukuran bunga, harga dan deskripsi. Untuk mempermudah pengguna mengenali *swipe* ke kanan atau ke kiri dalam melihat lebih banyak detail bunga, diberikan pula *indicator pagerview* di atas gambar bunga. Pada ujung kanan atas juga terdapat *icon sort* yang dapat digunakan untuk mengurutkan harga bunga mulai dari paling murah hingga paling mahal dan begitu juga sebaliknya.

d. Menampilkan halaman informasi tentang profil dan kontak meme florist

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman informasi tentang profil dan kontak Meme Florist ditunjukkan oleh , Gambar 6. 8, Gambar 6. 7, Gambar 6. 10, Gambar 6. 9, Gambar 6. 11, Gambar 6. 12, dan Gambar 6.13.



Gambar 6.2
Splashscreen aplikasi
Meme Florist



Gambar 6.1 Halaman
landing menu untuk
pengguna yang tidak *login*



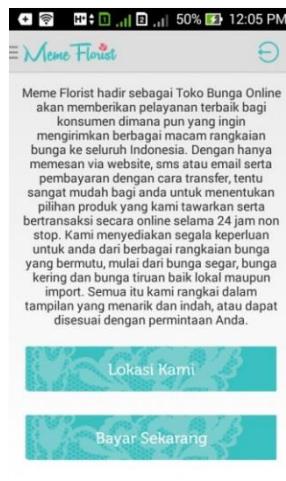
Gambar 6.4 Halaman
detail bunga



Gambar 6.3 Halaman
kategori bunga



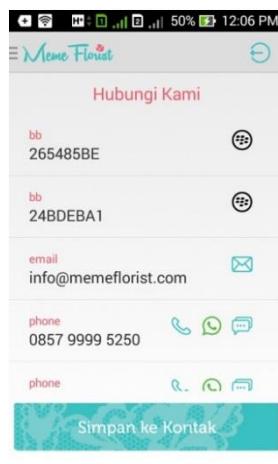
Gambar 6. 6 Halaman cara pembayaran



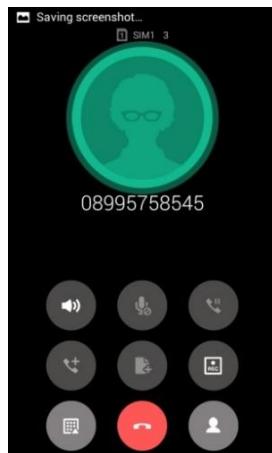
Gambar 6.5 Halaman tentang kami



Gambar 6. 7 Halaman lokasi kami



Gambar 6. 8 Halaman hubungi kami



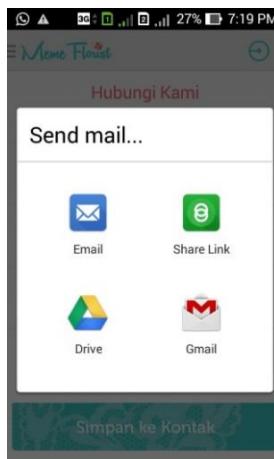
Gambar 6. 9 Halaman Direct call phone



Gambar 6. 10 Halaman Direct copied to clipboard



Gambar 6. 11 Halaman Direct to messaging



Gambar 6. 12 Halaman Direct to email app

Pada adalah halaman tentang kami yang memiliki tombol Lokasi Kami dan Bayar Sekarang. Ketika tombol Lokasi Kami ditekan, maka halaman akan berpindah ke halaman seperti pada Gambar 6. 7, sedangkan ketika tombol Bayar Sekarang ditekan akan berpindah halaman seperti pada Gambar 6. 5.

Pada Gambar 6. 8 adalah halaman yang menampilkan seluruh kontak Meme Florist dan terdapat tombol WhatsApp, BBM, telepon, SMS, *email* dan simpan ke kontak. Semua tombol tersebut dapat ditekan untuk berpindah ke halaman yang lain. Ketika tombol WhatsApp dan BBM ditekan, maka akan melakukan *copy to clipboard* seperti pada Gambar 6. 10. Ketika tombol telepon ditekan, maka halaman akan berpindah ke *call phone* seperti pada Gambar 6. 9. Ketika tombol SMS ditekan maka akan berpindah ke halaman *messaging* seperti pada Gambar 6. 11. Ketika tombol *email* ditekan, maka halaman akan berpindah ke *email app* seperti pada Gambar 6. 12. Ketika tombol Simpan ke Kontak ditekan, maka halaman akan berpindah ke *contact app* untuk menyimpan kontak Meme Florist seperti yang ditunjukkan pada Gambar 6.13.

e. Menampilkan halaman *login*

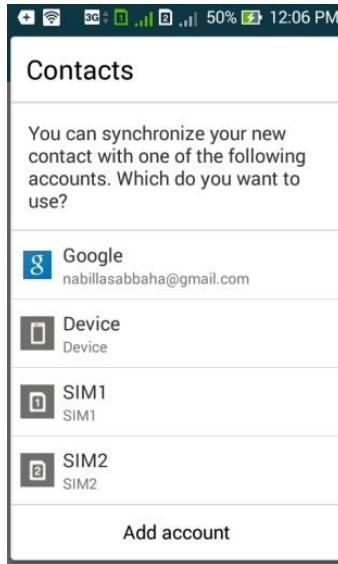
Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman *login* ditunjukkan oleh Gambar 6.14.

Melalui Gambar 6.14, dapat dilihat bahwa pengguna dapat menggunakan 2 jenis *login*, yaitu *login* menggunakan akun Google dan *login* menggunakan akun Meme Florist melalui *email* dan *password* yang telah terdaftar melalui registrasi. Pada halaman ini, terdapat tombol registrasi bagi pengguna yang ingin terdaftar dalam Meme Florist dan mendapatkan hak akses untuk dapat melakukan pemesanan *online*. Selain

itu juga terdapat tombol *forgot password* untuk pengguna melakukan permintaan *reset password*.

f. Menampilkan halaman *Landing Menu* pengguna yang telah *login*

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman *landing menu* pengguna yang telah *login* ditunjukkan oleh Gambar 6.15.



**Gambar 6.14 Halaman
*Direct to save to
contacts***

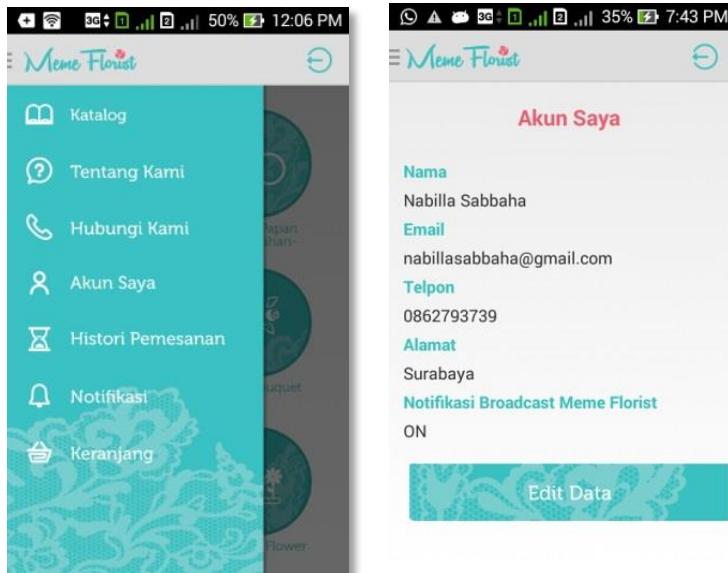


**Gambar 6.13 Halaman
*login pengguna***

Melalui Gambar 6.15, dapat dilihat bahwa pengguna mendapatkan 7 menu utama yang didapatkan pengguna yang telah login melalui aplikasi Meme Florist ini. Selain itu terdapat *icon logout* yang berada di ujung kanan atas ketika pengguna ingin *logout* dari aplikasi Meme Florist.

g. Menampilkan halaman profil pengguna

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan profil pengguna melalui halaman akun saya ditunjukkan oleh Gambar 6.16.



Gambar 6.15 Halaman *landing menu* bagi pengguna yang sudah login

Gambar 6.16 Halaman akun saya

h. Menampilkan halaman edit profil pengguna

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan edit profil pengguna melalui halaman edit akun saya ditunjukkan oleh Gambar 6.17.

Melalui Gambar 6.17, dapat dilihat bahwa pengguna dapat memperbarui data profil pengguna, melakukan pengaturan penerimaan notifikasi dengan *switch* notifikasi *broadcast* Meme Florist, dan menyimpan pengaturan tersebut dengan tombol Simpan Data.

i. Menampilkan halaman menambah bunga ke keranjang belanja

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menambah pesanan bunga ke keranjang belanja ditunjukkan oleh .

Melalui , dapat dilihat bahwa pengguna dapat menambah pesanan bunga ke keranjang belanja dengan mengisi beberapa *form* melalui halaman detil bunga seperti pada . dan menekan tombol Masukkan ke Keranjang. Untuk kotak dialog yang ditampilkan dalam menambah pesanan bunga ke keranjang belanja, pengguna perlu mengisi jumlah pembelian bunga yang dipilih dan mengisi pesan yang ingin disampaikan untuk bunga yang ingin dibeli tersebut.

j. Menampilkan halaman keranjang belanja

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan seluruh keranjang belanja yang dimiliki pengguna ditunjukkan oleh Gambar 6.19.

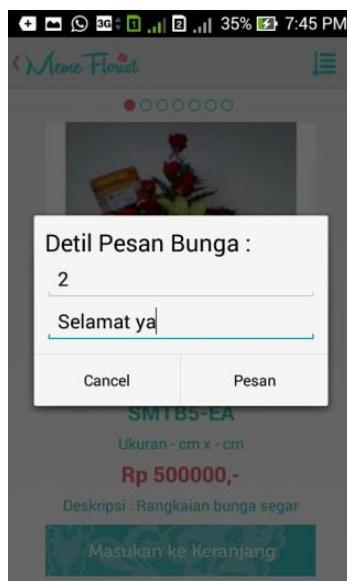
Melalui Gambar 6.19, dapat dilihat bahwa pengguna dapat melihat seluruh bunga yang sudah masuk ke dalam keranjang belanja. Disini pengguna dapat melakukan pengolahan keranjang belanja,

pengguna dapat menambah bunga dengan cara membuka menu katalog dan memilih kembali bunga yang ingin dibeli seperti pada Gambar 6.4. Kemudian menambah bunga ke dalam keranjang belanja seperti pada . Selain itu, pengguna dapat menghapus bunga yang sudah terlanjur masuk ke dalam keranjang belanja dengan menekan tombol silang (x) yang tersedia di ujung kanan atas masing-masing *item* bunga.



Gambar 6.18 Halaman edit akun saya

pemesanan
bunga
Hasil uji
coba dan evaluasi
yang telah dilakukan adalah aplikasi ini dapat



Gambar 6.17 Halaman menambah pesanan bunga ke keranjang belanja

menampilkan halaman pemesanan bunga ditunjukkan oleh Gambar 6.20.

Melalui Gambar 6.20, dapat dilihat bahwa pengguna dapat melakukan pemesanan bunga dengan mengisi data pengiriman bunga. Setelah, seluruh data yang dibutuhkan sudah terisi, maka pengguna dapat menekan tombol Bayar Sekarang untuk mendapatkan deskripsi cara pembayaran bunga yang telah dipesan tersebut seperti pada Gambar 6. 5.

l. Menampilkan halaman histori pemesanan

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman histori pemesanan ditunjukkan oleh Gambar 6. 22, Gambar 6. 22, dan Gambar 6. 23.

Melalui Gambar 6. 22, dapat dilihat bahwa pengguna dapat melihat histori pemesanan yang dimiliki pengguna. Kemudian jika pengguna ingin melihat detil histori pemesanan, pengguna hanya perlu melakukan event click atau menekan salah satu histori pemesanan yang ingin dilihat detilnya seperti yang ditunjukkan pada Gambar 6. 21, sedangkan yang akan ditampilkan adalah detil bunga yang dipesan pada histori pemesanan yang dipilih tersebut seperti yang ditunjukkan pada Gambar 6. 23.

m. Menampilkan halaman histori notifikasi

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman histori notifikasi ditunjukkan oleh Gambar 6.24.

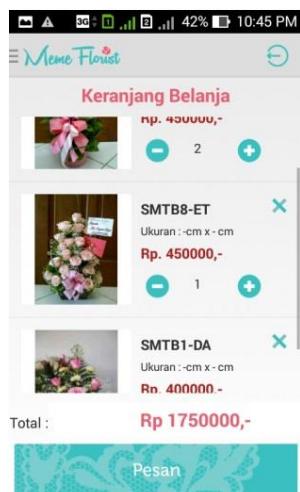
Melalui Gambar 6.24, dapat dilihat bahwa pengguna dapat melihat histori seluruh notifikasi yang dimiliki pengguna yang berasal dari Meme Florist. Histori notifikasi ini hanya menyimpan histori notifikasi *broadcast* yang diberikan Meme Florist

biasanya bersifat umum mengenai promosi produk baru atau diskon.

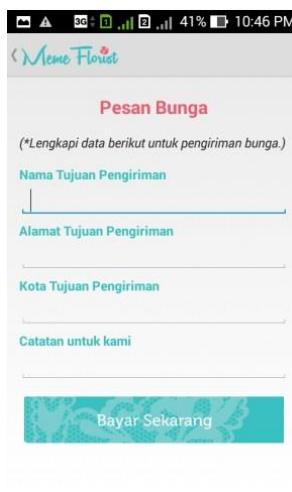
n. Menampilkan halaman registrasi

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman registrasi ditunjukkan oleh Gambar 6. 26.

Melalui Gambar 6. 26, dapat dilihat bahwa pengguna dapat melakukan registrasi dengan mengisi form registrasi kemudian jika semua data telah terisi pengguna dapat menekan tombol buat akun baru. Jika proses registrasi berjalan dengan sukses, maka antarmuka selanjutnya adalah halaman *login* yang akan menampilkan *toast* untuk pengguna melakukan cek email untuk konfirmasi akun baru seperti yang ditunjukkan pada Gambar 6. 25.



Gambar 6.20 Halaman keranjang belanja



Gambar 6.19 Halaman pemesanan bunga



Gambar 6. 22 Event click pada histori pemesanan



Gambar 6. 21 Halaman histori pemesanan



Gambar 6. 24 Halaman detail histori pemesanan



Gambar 6.23 Halaman histori notifikasi

o. Menampilkan halaman *forgot password*

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan halaman histori notifikasi ditunjukkan oleh Gambar 6. 28.

Melalui Gambar 6. 28, dapat dilihat bahwa pengguna dapat melakukan permintaan *reset password* melalui halaman *forgot password* dengan mengisi *form* yang disediakan dengan *email* yang digunakan untuk *login*. Setelah itu, pengguna dapat menekan tombol kirim *email* untuk mendapatkan *email* konfirmasi *reset password* dari administrator Meme Florist dan prosesnya seperti yang ditunjukkan pada Gambar 6. 27.

p. Menampilkan *push notification broadcast* dari Meme Florist

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan *push notification broadcast* dari Meme Florist ditunjukkan oleh Gambar 6. 29.

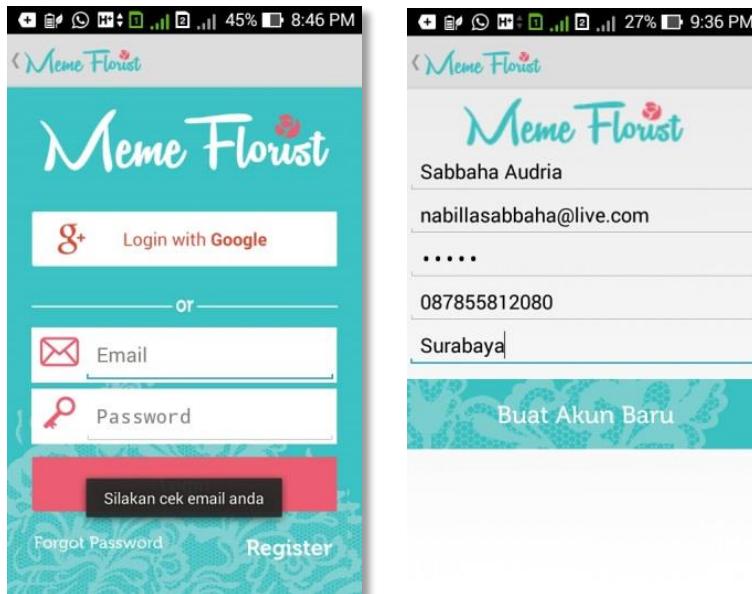
Melalui Gambar 6. 29, dapat dilihat bahwa pengguna mendapatkan notifikasi dari Meme Florist yang muncul di *Notification Bar device*. Ketika dilakukan *event click* atau menekan notifikasi di *Notification Bar device*, maka aplikasi Meme Florist akan otomatis dijalankan dan muncul toast pesan notifikasi yang dikirimkan seperti yang ditunjukkan pada gambar Gambar 6. 30.

q. Menampilkan *push notification personal pengiriman bunga* dari Meme Florist

Hasil uji coba dan evaluasi yang telah dilakukan adalah aplikasi ini dapat menampilkan *push notification personal* dari Meme Florist mengenai status pengiriman bunga yang dipesan ditunjukkan oleh Gambar 6. 32.

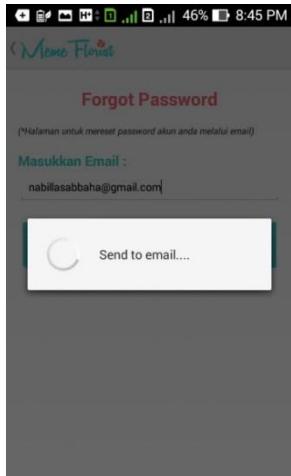
Melalui Gambar 6. 32, dapat dilihat bahwa pengguna mendapatkan notifikasi dari Meme Florist

yang muncul di *Notification Bar device*. Ketika dilakukan *event click* atau menekan notifikasi di *Notification Bar device*, maka aplikasi Meme Florist akan otomatis dijalankan dan menampilkan antarmuka notifikasi pengiriman bunga seperti yang ditunjukkan pada gambar Gambar 6. 31. Setelah itu pengguna dapat melakukan cek status pengiriman bunga pada histori pemesanan seperti yang ditunjukkan pada Gambar 6. 22.



Gambar 6. 26 Halaman login sebagai bukti sukses melakukan registrasi

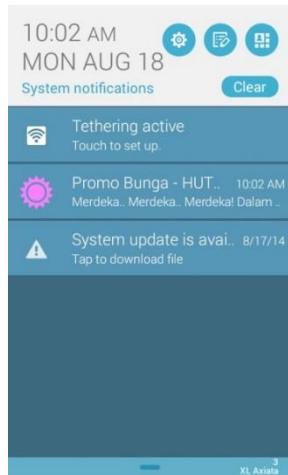
Gambar 6. 25 Halaman registrasi



Gambar 6. 28 Halaman proses permintaan reset password



Gambar 6. 27 Halaman forgot password



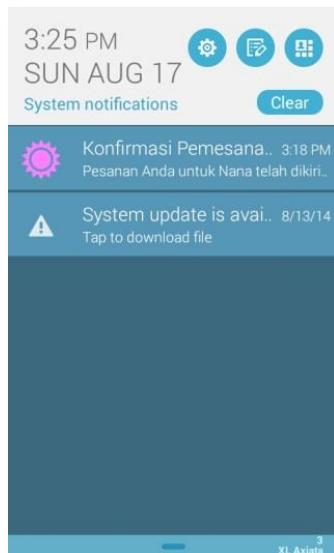
**Gambar 6. 30 Antarmuka
Notification Bar device untuk
notifikasi broadcast**



**Gambar 6. 29 Toast isi pesan
notifikasi**



Gambar 6. 32 Halaman notifikasi pengiriman



Gambar 6. 31 Antarmuka Notification Bar device untuk notifikasi personal

6.2 Aplikasi Web Administrator Meme Florist

Pada subbab ini, akan dijelaskan mengenai proses ujicoba pada aplikasi web administrator untuk setiap fungsionalitas Meme Florist.

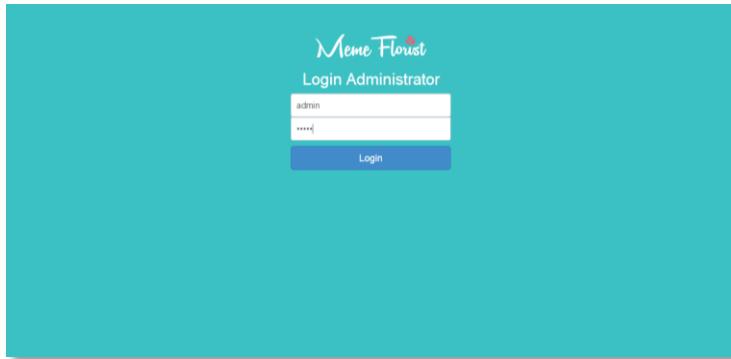
6.2.1 Lingkungan Uji Coba

Lingkungan Uji Coba yang digunakan dalam proses pengembangan aplikasi web Meme Florist ini adalah Microsoft Windows 8, Java Runtime Environment 7, dan perangkat pengembang yang digunakan adalah Eclipse Kepler 4.3.0, Google App Engine 1.9.6, Java Development Kit 1.7.0_60 dan peramban Google Chrome atau Mozilla Firefox.

6.2.2 Menampilkan Hasil Uji Coba

a. Menampilkan halaman *Login* administrator

Dapat dilihat pada Gambar 6.33 bahwa administrator diharuskan melakukan *login* sebelum dapat menggunakan fitur-fitur di dalamnya melalui halaman *login* administrator Meme Florist. Caranya yakni administrator diminta untuk memasukkan *username* dan *password* yang valid. Setelah tombol *login* diklik dan jika *login* sukses, administrator akan diarahkan menuju halaman beranda.



Gambar 6.33 Halaman *Login* administrator

b. Menampilkan halaman beranda Meme Florist

Setelah administrator melakukan *login* serta *username* dan *password* yang dimasukkan dinyatakan valid, maka administrator akan diarahkan menuju ke halaman beranda aplikasi *web* Meme Florist seperti yang ditunjukkan oleh Gambar 6.34.

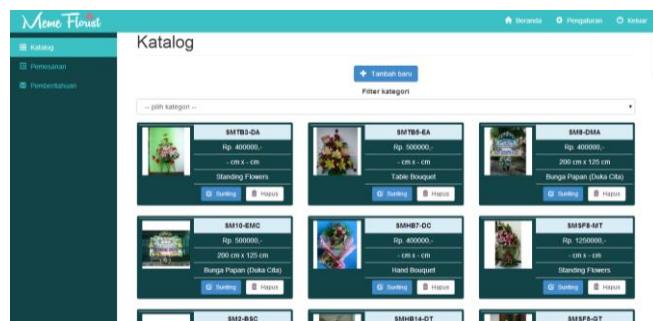
c. Menampilkan halaman daftar katalog bunga

Dalam pengelolaan katalog bunga tentunya dibutuhkan halaman untuk melihat daftar katalog dari

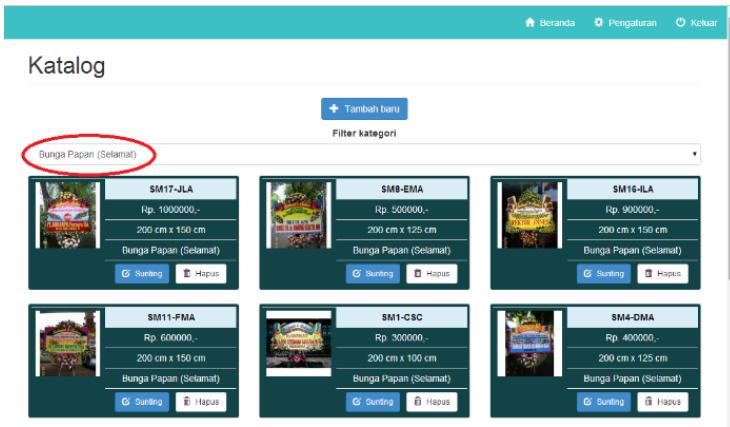
bunga. Di halaman hasil uji coba dan evaluasi yang ditunjukkan oleh Gambar 6.35, administrator dapat melihat daftar katalog bunga yang dapat dikelolanya. Selain itu, administrator juga dapat melihat daftar katalog bunga berdasarkan kategori yang dipilih dengan menggunakan fitur *filter* kategori yang ditunjukkan oleh Gambar 6.36.



Gambar 6.34 Halaman beranda web administrator



Gambar 6.35 Halaman daftar katalog bunga



SKU	Nama	Harga	Dimensi	Kategori
SM17-JLA	Bunga Papan (Selamat)	Rp. 1000000,-	200 cm x 150 cm	Bunga Papan (Selamat)
SMB-EMA	Bunga Papan (Selamat)	Rp. 500000,-	200 cm x 125 cm	Bunga Papan (Selamat)
SM16-ILA	Bunga Papan (Selamat)	Rp. 900000,-	200 cm x 150 cm	Bunga Papan (Selamat)
SM11-FMA	Bunga Papan (Selamat)	Rp. 600000,-	200 cm x 150 cm	Bunga Papan (Selamat)
SM1-CSC	Bunga Papan (Selamat)	Rp. 300000,-	200 cm x 100 cm	Bunga Papan (Selamat)
SM4-DMA	Bunga Papan (Selamat)	Rp. 400000,-	200 cm x 125 cm	Bunga Papan (Selamat)

Gambar 6.36 Halaman daftar katalog bunga setelah filter kategori

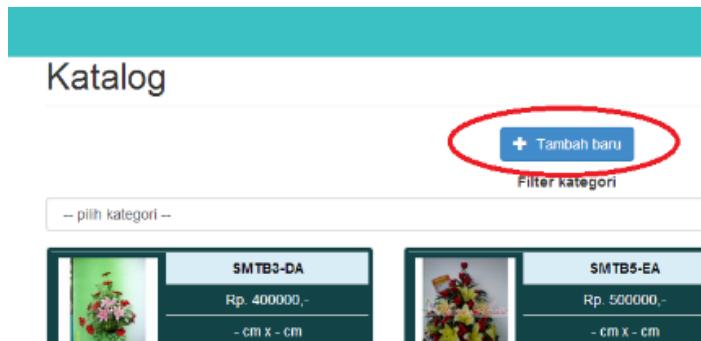
d. Menampilkan Halaman Menambah Data Katalog Bunga

Administrator dapat menambahkan data katalog bunga melalui halaman melihat daftar katalog bunga seperti yang ditunjukkan pada Gambar 6.37. Kemudian di halaman penambahan data katalog bunga, administrator mengisikan data-data yang diperlukan. Kemudian setelah data dirasa sudah lengkap, administrator memilih tombol Simpan seperti yang ditunjukkan pada Gambar 6.38.

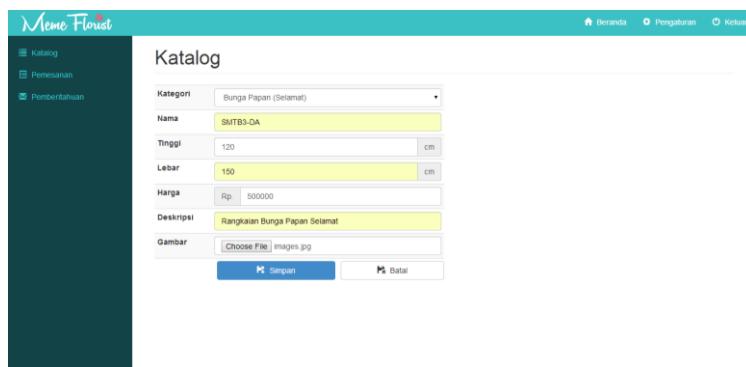
e. Menampilkan Halaman Mengubah Data Katalog Bunga

Dalam pengelolaan data katalog bunga, kadang kala diperlukan perubahan data seperti harga, foto bunga, ataupun yang lain. Administrator dapat mengubah data katalog bunga yang dimaksud dengan

terlebih dahulu memilih bunga yang akan diubah melalui halaman daftar katalog bunga lalu memilih tombol Sunting seperti yang ditunjukkan oleh Gambar 6.39. Kemudian administrator dapat mengisi data isian yang akan diubah seperti yang tampak pada Gambar 6.40, setelah selesai lalu diklik tombol *Update*.



Gambar 6.37 Tombol tambah di halaman daftar katalog bunga

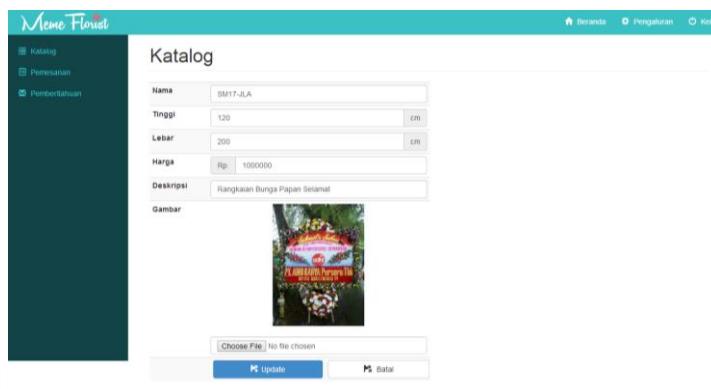


Kategori	Bunga Papan Selamat
Nama	SMTB3-DA
Tinggi	120 cm
Lebar	150 cm
Harga	Rp. 500000,-
Deskripsi	Rangkaian Bunga Papan Selamat
Gambar	<input type="file"/> images.png
<input type="button" value="Simpan"/> <input type="button" value="Batal"/>	

Gambar 6.38 Halaman menambah data katalog bunga



Gambar 6.39 Tombol Sunting untuk mengubah data katalog bunga



Gambar 6.40 Halaman mengubah data katalog bunga

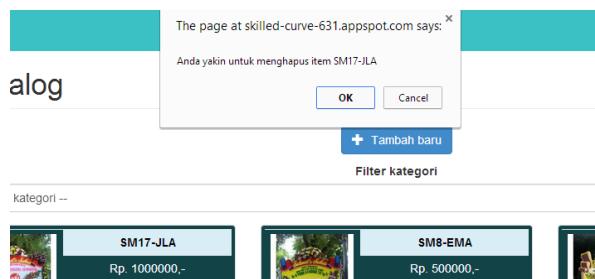
f. Menampilkan Halaman Menghapus Data Katalog Bunga

Administrator dapat melakukan penghapusan data katalog bunga dengan memilihnya terlebih dahulu, kemudian memilih tombol Hapus seperti pada uji coba dan evaluasi yang ditunjukkan oleh Gambar 6.41. Kemudian administrator akan diperlihatkan dengan

confirm dialog untuk memastikan apakah administrator yakin untuk menghapus data katalog bunga yang dipilih. Uji coba dan evaluasi ini ditunjukkan pada Gambar 6.42.



Gambar 6.41 Tombol hapus untuk menghapus data katalog bunga

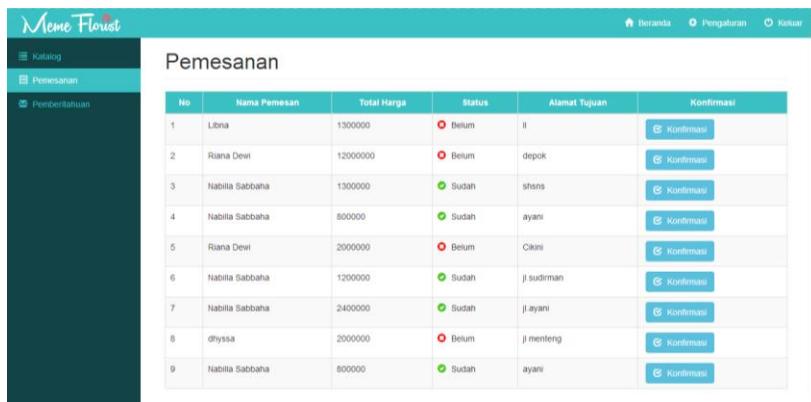


Gambar 6.42 Confirm Dialog untuk menghapus katalog yang dipilih

g. Menampilkan Halaman Daftar Pemesanan Pelanggan

Data pemesanan pelanggan yang telah disimpan dapat dilihat oleh administrator agar dapat diproses. Hasil uji coba dan evaluasi untuk melihat daftar

pemesanan pelanggan ini ditunjukkan oleh Gambar 6.43.



The screenshot shows a mobile application interface for 'Meme Florist'. The top navigation bar includes links for 'Beranda', 'Pengaturan', and 'Ketar'. On the left, there's a vertical sidebar with three menu items: 'Katalog', 'Pemesanan' (which is currently selected and highlighted in blue), and 'Pemberitahuan'. The main content area is titled 'Pemesanan' and displays a table of 9 pending orders. The columns in the table are: 'No', 'Nama Pemesan', 'Total Harga', 'Status', 'Alamat Tujuan', and 'Konfirmasi'. Each row contains a row number, the name of the customer, the total price, the status (either 'Belum' or 'Sudah'), the delivery address, and a blue button labeled 'Konfirmasi'.

No	Nama Pemesan	Total Harga	Status	Alamat Tujuan	Konfirmasi
1	Libna	1300000	● Belum	jl	
2	Riana Dewi	12000000	● Belum	depok	
3	Nabilla Sabbaha	1300000	● Sudah	stans	
4	Nabilla Sabbaha	800000	● Sudah	ayani	
5	Riana Dewi	2000000	● Belum	Cikini	
6	Nabilla Sabbaha	1200000	● Sudah	jl sudirman	
7	Nabilla Sabbaha	2400000	● Sudah	jl ayani	
8	dhyssa	2000000	● Belum	jl menteng	
9	Nabilla Sabbaha	800000	● Sudah	ayani	

Gambar 6.43 Halaman daftar pemesanan pelanggan

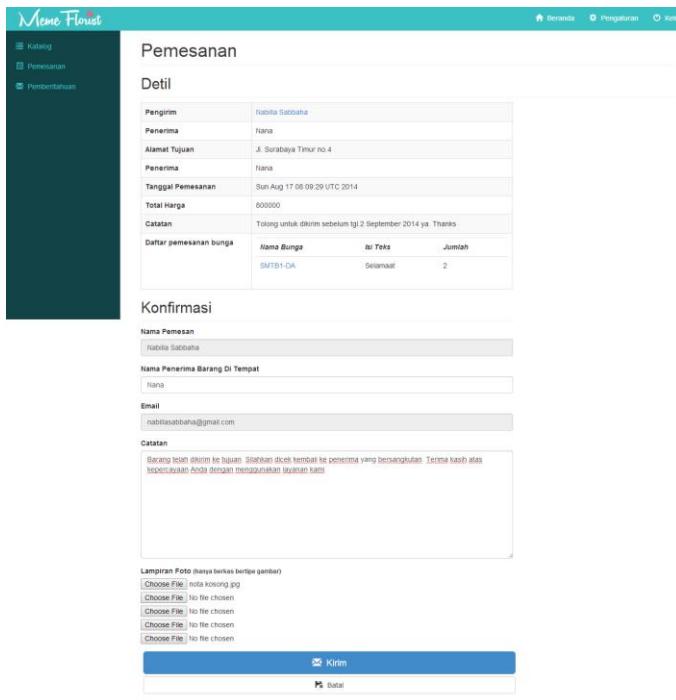
h. Menampilkan Konfirmasi Pemesanan Pelanggan

Pemesanan yang telah diproses atau dikirim ke tempat tujuan, akan dikonfirmasi oleh administrator kepada pelanggan bahwa pemesanannya telah selesai dikirim. Administrator dapat melakukan konfirmasi pemesanan pelanggan seperti pada hasil uji coba dan evaluasi yang ditunjukkan oleh Gambar 6.44. Administrator akan mengisi field yang dibutuhkan, lalu memilih tombol Kirim untuk mengirimkan konfirmasi berupa *push notification* ke *device* dan *email* pelanggan yang bersangkutan seperti pada Gambar 6.45. Lalu, pelanggan akan menerima *email* seperti Gambar 6.46 dan *push notification* seperti pada Gambar 6.48 dan Gambar 6.49. Pemesanan yang telah

dikonfirmasi akan berubah statusnya seperti yang ditunjukkan pada Gambar 6.47.

4	Nabilia Sabbaha	600000	<input checked="" type="checkbox"/> Sudah	ayani	<input checked="" type="button"/> Konfirmasi
5	Nabilia Sabbaha	800000	<input type="checkbox"/> Belum	Jl. Surabaya Timur no.4	<input checked="" type="button"/> Konfirmasi
6	Riana Dewi	2000000	<input type="checkbox"/> Belum	Cikini	<input checked="" type="button"/> Konfirmasi

Gambar 6.44 Salah satu pemesanan pelanggan sebelum dikonfirmasi



The screenshot shows the 'Pemesanan' (Order) screen of the Meme Florist app. On the left, there's a sidebar with navigation options: Katalog, Pemesanan, and Pembentahan. The main area is titled 'Pemesanan' and contains two sections: 'Detil' and 'Konfirmasi'.

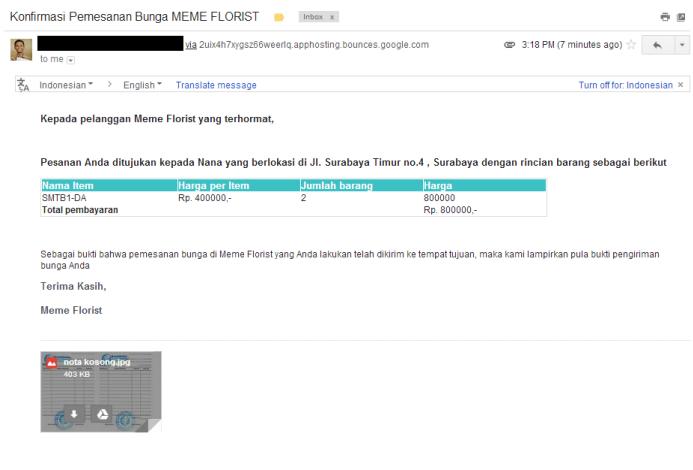
Detil:

Pengirim	Nabilia Sabbaha						
Penerima	Nana						
Alamat Tujuan	Jl. Surabaya Timur no.4						
Penerima	Nana						
Tanggal Pemesanan	Sun Aug 17 08:09:29 UTC 2014						
Total Harga	800000						
Catatan	Tolong untuk dikirim sebelum tgl 2 September 2014 ya. Thanks						
Daftar pemesanan bunga	<table border="1"> <thead> <tr> <th>Nama Bunga</th> <th>Bu Teks</th> <th>Jumlah</th> </tr> </thead> <tbody> <tr> <td>SM161-DA</td> <td>Selamat</td> <td>2</td> </tr> </tbody> </table>	Nama Bunga	Bu Teks	Jumlah	SM161-DA	Selamat	2
Nama Bunga	Bu Teks	Jumlah					
SM161-DA	Selamat	2					

Konfirmasi:

Fields for Name of Sender (Nabilia Sabbaha), Name of Receiver (Nana), Email (nabilasabbaha@gmail.com), and Notes (Catatan) are filled. The notes mention a deadline of August 2, 2014. At the bottom, there are fields for attachments ('Lampiran Foto') and a large blue 'Kirim' (Send) button.

Gambar 6.45 Halaman konfirmasi pemesanan pelanggan



Gambar 6.46 Email konfirmasi pemesanan bunga

4	Nabila Sabbathia	800000	<input checked="" type="checkbox"/> Sudah	ayani	Konfirmasi
5	Nabila Sabbathia	800000	<input checked="" type="checkbox"/> Sudah	Jl. Surabaya Timur no.4	Konfirmasi
6	Riana Dewi	2000000	<input type="checkbox"/> Belum	Cikni	Konfirmasi

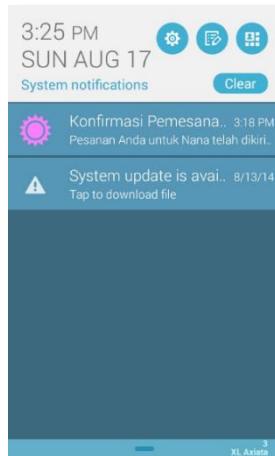
Gambar 6.47 Salah satu pemesanan pelanggan setelah dikonfirmasi

i. Menampilkan Pengiriman Pemberitahuan Kepada Pelanggan

Administrator dapat mengirim pemberitahuan berupa *push notification* berupa *push notification* kepada pelanggan melalui halaman pengiriman halaman pengiriman pemberitahuan kepada pelanggan seperti yang ditunjukkan oleh

Gambar 6.52. Setelah administrator mengisi *field* yang diperlukan dan memilih tombol Kirim, maka sesuai hasil uji coba dan evaluasi, pelanggan akan

menerima *push notification* di *device* seperti yang ditunjukkan pada Gambar 6. 50 dan Gambar 6. 51.



Gambar 6. 49 Notifikasi konfirmasi pemesanan di *device* pelanggan



Gambar 6. 48 Detail notifikasi konfirmasi pengiriman di *device* pelanggan



Gambar 6. 51 Notifikasi pemberitahuan di device pelanggan

Gambar 6. 50 Histori notifikasi yang masuk ke device pelanggan

Gambar 6.52 Halaman pengiriman pemberitahuan ke pelanggan

j. Menampilkan Pengubahan Profil dan Kontak Toko Meme Florist

Profil dan kontak toko Meme Florist yang ditampilkan melalui aplikasi Meme Florist dapat diubah melalui halaman pengaturan profil dan kontak di

aplikasi *web* Meme Florist. Administrator terlebih dahulu akan ditampilkan halaman yang berisi profil dan kontak Meme Florist seperti yang ditunjukkan pada Gambar 6.53. Kemudian, administrator memilih *Hyperlink* dengan tulisan “Edit” dan administrator dapat mengubah data profil dan toko Meme Florist seperti yang ditunjukkan pada Gambar 6.54. Setelah selesai, maka dapat dipilih tombol Simpan.

k. Menampilkan Aktivasi Akun Pelanggan

Terlebih dahulu pelanggan diminta untuk melakukan registrasi pada aplikasi Meme Florist seperti pada Gambar 6. 55. Kemudian, pelanggan akan menerima *email* berupa *link* seperti pada Gambar 6. 56 yang akan mengarahkan pelanggan tersebut untuk melakukan aktivasi. Setelah *link* tersebut dituju, maka akan ditampilkan halaman bahwa aktivasi akun pelanggan telah berhasil seperti yang ditunjukkan pada Gambar 6.57. Jika *link* yang dimasukkan salah, maka akan gagal seperti tampak pada Gambar 6.58.

Meme Florist

Katalog
Pemesanan
Pembentahan

Beranda

Pengaturan
Edit

Username	admin
Password	*****
Lokasi	 <div style="text-align: right; margin-top: 5px;"> Latitude : -6.96666670 </div> <div style="text-align: right; margin-top: 5px;"> Longitude : 110.41666670 </div>
Alamat	Kota Semarang
Tentang	<p>Meme Florist hadir sebagai Toko Bunga Online akan memberikan pelayanan terbaik bagi konsumen dimana pun yang ingin mengirimkan berbagai macam rangkaian bunga ke seluruh Indonesia. Dengan hanya memesan via website, sms atau email serta pembayaran dengan cara transfer, tentu sangat mudah bagi anda untuk menentukan pilihan produk yang kami tawarkan serta bertransaksi secara online selama 24 jam non stop. Kami menyediakan segala keperluan untuk anda dari berbagai rangkaian bunga yang bermutu, mulai dari bunga segar, bunga kering dan bunga tiruan baik lokal maupun import. Semua itu kami rangkai dalam tampilan yang menarik dan indah, atau dapat disesuaikan dengan permintaan Anda.</p>

Tipe kontak	No
bb	265485BE
bb	24BDEBA1
email	info@meme florist.com
phone	0857 9999 5250
phone	0899 5758 545

Nama Bank	Atas Nama	No. Rekening	Logo Bank
Bank Mandiri	a/m Mariani	135-000-000-5478	<button style="border: none; background-color: #0070C0; color: white; padding: 2px 5px; font-size: 0.8em;">Lihat</button>
BCA	a/m Mariani	009-517-8354	<button style="border: none; background-color: #0070C0; color: white; padding: 2px 5px; font-size: 0.8em;">Lihat</button>

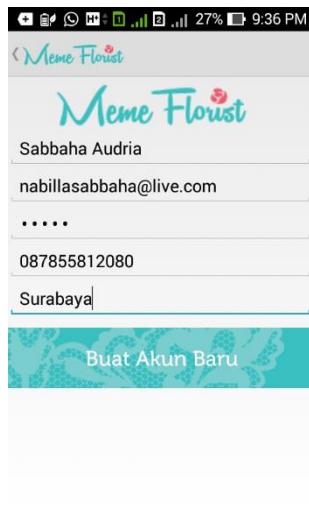
Gambar 6.53 Halaman pengaturan profil dan kontak Meme Florist

Beranda
Pengaturan
Keluar

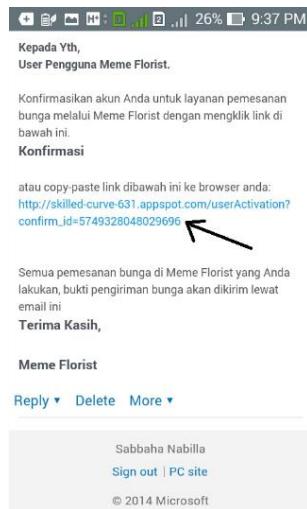
Pengaturan

Username																																																																																												
admin																																																																																												
Password																																																																																												
Info: Double click untuk mengganti lokasi <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> Cari tempat <input style="width: 200px; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; margin-right: 10px;" type="text"/> Caril </div> <div style="margin-top: 10px;"> Latitude <input style="width: 200px; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; margin-bottom: 5px;" type="text" value="-6.96666670"/> Longitude <input style="width: 200px; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px;" type="text" value="110.41666670"/> </div>																																																																																												
Alamat	<input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px;" type="text" value="Kota Semarang"/>																																																																																											
Tentang	<p>Meme Florist hadir sebagai Toko Bunga Online akan memberikan pelayanan terbaik bagi konsumen dimana pun yang ingin mengirimkan berbagai macam rangkaian bunga ke seluruh Indonesia. Dengan hanya memesan via website, sms atau email serta pembayaran dengan cara transfer, tentu sangat mudah bagi anda untuk menentukan pilihan produk yang kami tawarkan serta bertransaksi secara online selama 24 jam non stop. Kami menyediakan segala keperluan untuk anda dari berbagai rangkaian bunga yang bermutu, mulai dari bunga segar, bunga kering dan bunga tiruan baik lokal maupun import. Semua itu kami rangkai dalam tampilan yang menarik dan indah, atau dapat disesuaikan dengan permintaan Anda.</p>																																																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">accountNumber</td> <td>Bank</td> <td>Bank Mandiri</td> <td>a/m</td> <td>a/m Mariani</td> <td>No. Rek.</td> <td>135-000-000-5471</td> <td>Logo Bank</td> <td>Lihat</td> <td>Ubah</td> <td style="width: 10px;">-</td> <td style="width: 10px;">+</td> </tr> <tr> <td>accountNumber</td> <td>Bank</td> <td>BCA</td> <td>a/m</td> <td>a/m Mariani</td> <td>No. Rek.</td> <td>009-517-8354</td> <td>Logo Bank</td> <td>Lihat</td> <td>Ubah</td> <td>-</td> <td>+</td> </tr> <tr> <td>tb</td> <td colspan="10">265485BE</td> <td>-</td> <td>+</td> </tr> <tr> <td>tb</td> <td colspan="10">24BDEBA1</td> <td>-</td> <td>+</td> </tr> <tr> <td>email</td> <td colspan="10">info@menneflorist.com</td> <td>-</td> <td>+</td> </tr> <tr> <td>phone</td> <td colspan="10">0857 9999 5250</td> <td>-</td> <td>+</td> </tr> <tr> <td>phone</td> <td colspan="10">0899 5758 545</td> <td>-</td> <td>+</td> </tr> </table>				accountNumber	Bank	Bank Mandiri	a/m	a/m Mariani	No. Rek.	135-000-000-5471	Logo Bank	Lihat	Ubah	-	+	accountNumber	Bank	BCA	a/m	a/m Mariani	No. Rek.	009-517-8354	Logo Bank	Lihat	Ubah	-	+	tb	265485BE										-	+	tb	24BDEBA1										-	+	email	info@menneflorist.com										-	+	phone	0857 9999 5250										-	+	phone	0899 5758 545										-	+
accountNumber	Bank	Bank Mandiri	a/m	a/m Mariani	No. Rek.	135-000-000-5471	Logo Bank	Lihat	Ubah	-	+																																																																																	
accountNumber	Bank	BCA	a/m	a/m Mariani	No. Rek.	009-517-8354	Logo Bank	Lihat	Ubah	-	+																																																																																	
tb	265485BE										-	+																																																																																
tb	24BDEBA1										-	+																																																																																
email	info@menneflorist.com										-	+																																																																																
phone	0857 9999 5250										-	+																																																																																
phone	0899 5758 545										-	+																																																																																
Simpan Batal																																																																																												

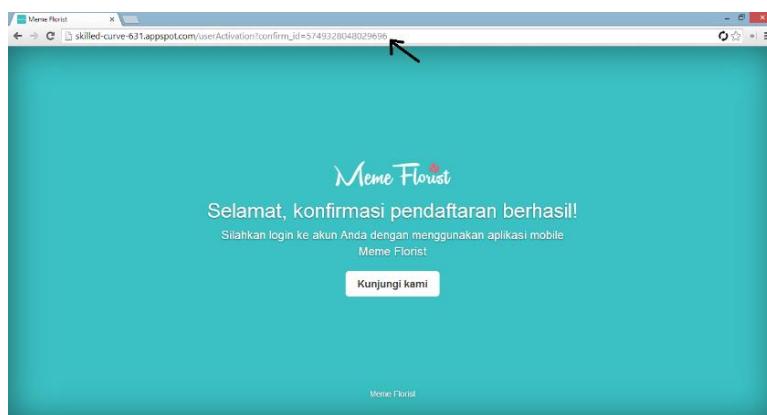
Gambar 6.54 Halaman mengubah data profil dan kontak Meme Florist



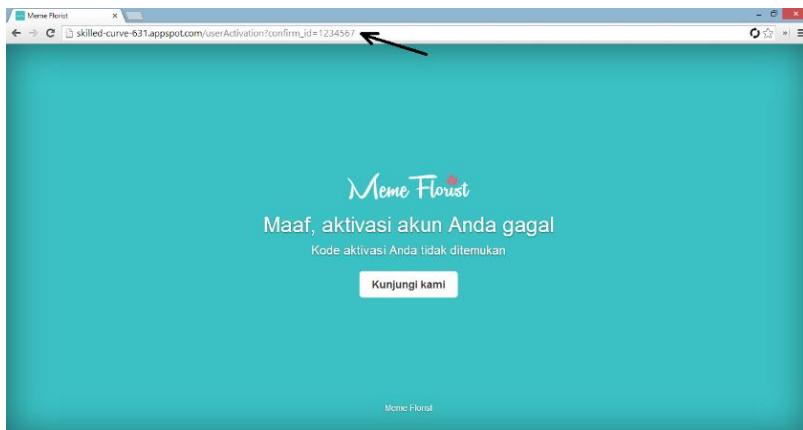
Gambar 6. 56 Registrasi pelanggan baru di aplikasi Meme Florist



Gambar 6. 55 Email link aktivasi pelanggan baru



Gambar 6.57 Halaman notifikasi aktivasi akun berhasil

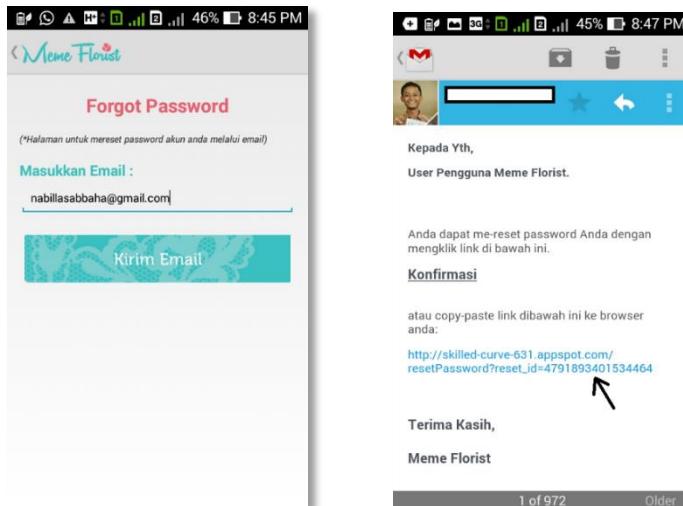


Gambar 6.58 Halaman notifikasi aktivasi akun gagal

I. Menampilkan *Reset Password* Akun Pelanggan

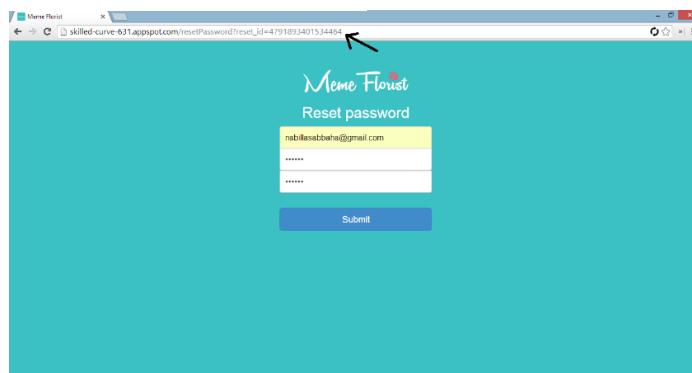
Terlebih dahulu pelanggan diminta untuk mengirimkan permintaan *reset password* pada aplikasi Meme Florist dengan fitur *forgot password* seperti pada Gambar 6. 59. Kemudian, pelanggan akan menerima *email* berupa *link* seperti pada Gambar 6. 60 yang akan mengarahkan pelanggan tersebut untuk mengganti *password* yang lama dengan yang baru. Setelah *link* tersebut dituju, maka akan ditampilkan halaman untuk diisi oleh pelanggan berupa *email* dan *password* baru seperti yang ditunjukkan pada Gambar 6.61. Setelah *password* dinyatakan telah berhasil diganti, pelanggan akan diarahkan ke halaman notifikasi bahwa penggantian *password* telah sukses seperti ditunjukkan

oleh Gambar 6.62. Jika *link* yang digunakan tidak valid, maka akan tampak seperti pada Gambar 6.63.

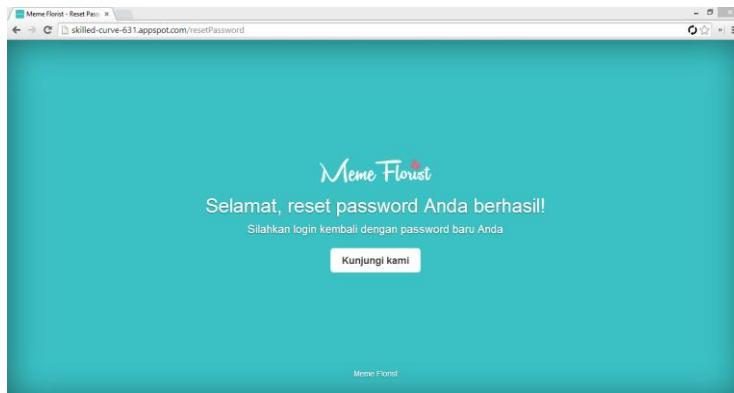


Gambar 6. 60 Permintaan *forgot password* di aplikasi Meme Florist

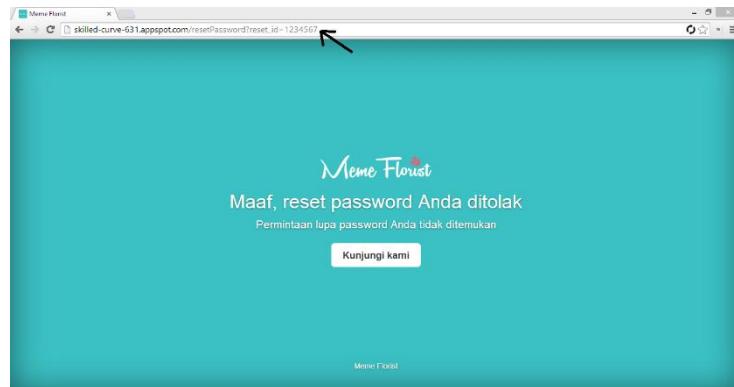
Gambar 6. 59 Email berisi *link* untuk *reset password*



Gambar 6.61 Halaman untuk *reset password*



Gambar 6.62 Halaman notifikasi *reset password* berhasil



Gambar 6.63 Halaman notifikasi permintaan *reset password* ditolak

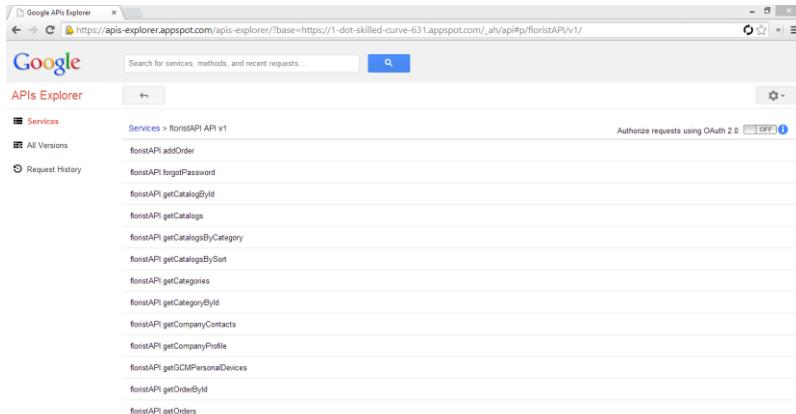
m. **API (*Application Programming Interface*) Florist Endpoint**

Untuk uji coba dan evaluasi dari Florist API sendiri langsung diimplementasikan melalui aplikasi Meme Florist. Dengan adanya API ini, pengembangan

aplikasi Android menjadi lebih mudah, karena seperti penggunaan *library* yang ditambahkan ke dalam *Android project* dan *developer* cukup dengan memanggil fungsi-fungsi di dalamnya. Namun untuk melihat API yang terdaftar dapat dilihat melalui Google APIs Explorer seperti yang ditunjukkan pada Gambar 6.64. Kemudian daftar fungsi yang dapat diakses seperti pada Gambar 6.65.



Gambar 6.64 Halaman menampilkan daftar API di Google APIs explorer



Gambar 6.65 Halaman menampilkan daftar fungsi dari Florist API

BAB VII

KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dan saran dalam pengembangan aplikasi Meme Florist dengan Google App Engine berbasis perangkat *mobile* Android untuk pemesanan bunga *online*.

7.1. Kesimpulan

Berdasarkan implementasi kode program yang dilakukan, menghasilkan ujicoba dan evaluasi yang baik dan sudah memenuhi analisis dan perancangan sistem aplikasi perangkat mobile dan aplikasi web administrator Meme Florist.

Melalui aplikasi perangkat *mobile* maupun aplikasi *web* administrator Meme Florist, seluruh kasus penggunaan telah memenuhi analisis kebutuhan pengguna akan aplikasi pemesanan bunga *online*. Selain itu, aplikasi perangkat mobile dan aplikasi *web* Meme Florist telah memenuhi tujuan pembuatan aplikasi seperti berikut ini:

- Pengguna dapat memilih bunga melalui katalog dan melakukan pemesanan bunga secara *online* dengan kuantitas tertentu.
- Pengguna dapat menerima notifikasi pengiriman bunga secara *personal* dan notifikasi promosi harga bunga.
- Pengguna dapat melakukan pengaturan penerimaan notifikasi
- Administrator dapat memasukkan data produk melalui *website*.
- Administrator dapat memantau data pemesanan bunga pengguna.
- Administrator dalam memberikan notifikasi pengiriman bunga secara *personal* dan notifikasi promosi harga bunga pada pengguna.
- Administrator dapat mengatur informasi tentang profil dan kontak meme florist
- Administrator dapat memberikan konfirmasi aktivasi akun, konfirmasi permintaan *password*, dan konfirmasi pemesanan

Secara keseluruhan, aplikasi Meme Florist dengan Google App Engine berbasis perangkat *mobile* Android untuk pemesanan bunga *online* dapat memberikan kemudahan bagi seorang pelanggan toko bunga Meme Florist dalam melakukan pembelian bunga, mengingat saat ini pengguna perangkat mobile Android semakin tinggi jumlahnya, sehingga transaksi pembelian bunga dapat lebih praktis dan mudah dilakukan dimana saja dan kapan saja. Begitu pula dengan Administrator Meme Florist, yang dapat kapanpun dan dimanapun dapat memperbarui data terkait produk bunga karena *web* yang dibangun memanfaatkan bootstrap untuk antarmuka aplikasi sehingga memiliki antarmuka yang *responsive* terhadap *device* yang digunakan.

7.2. Saran

Dalam pengembangan aplikasi perangkat *mobile* Meme Florist, seluruh datanya masih disimpan di *cloud* Google App Engine, sehingga setiap mengakses data terkait Meme Florist harus *retrieve* data dengan melakukan koneksi internet. Sebaiknya dalam pengembangan aplikasi perangkat bergerak Meme Florist selanjutnya, data tidak seluruhnya *retrieve* dari *cloud*, tapi disimpan di *cache*, sehingga ketika pengguna tidak memiliki koneksi internet masih bisa mengakses aplikasi Meme Florist. Namun jika ingin mendapatkan data terbaru, pengguna yang memiliki koneksi internet dapat melakukan proses *refresh* data sehingga aplikasi Meme Florist dapat melakukan *retrieve* data dari Google App Engine untuk memperbarui data *cache* pada *device*.

Untuk aplikasi *web* administrator Meme Florist untuk kedepannya diharapkan administrator Meme Florist dapat secara berkala memperbarui data terkait produk bunga maupun data informasi terkait Meme Florist. Hal ini bertujuan untuk memperbarui pula data yang diakses oleh pengguna pada perangkat *mobile* sehingga selalu mendapatkan data terbaru mengenai Meme Florist.

DAFTAR PUSTAKA

- [1] “Tentang 7Langit” [Online]. Tersedia: <http://www.7Langit.com/about/> [diakses 10 Agustus 2014].
- [2] Arfiandwi, O. [23 Juni 2014). Deskripsi proses bisnis yang diinginkan klien. [N. Sabbaha, & A. Baskara, Pewawancara].
- [3] “Klien 7Langit” [Online] Tersedia : <http://7langit.com/clients/> [diakses 10 Agustus 2014].
- [4] Arfiandwi, O. [23 Juni 2014). Sejarah 7Langit. [N. Sabbaha, & A. Baskara, Pewawancara].
- [5] “Sejarah Perkembangan Android” [Online]. Tersedia: <http://www.tekno-pedia.com/keunikan-nama-versi-os-Android-dan-sejarah-perkembangannya/> [diakses 19 Agustus 2014].
- [6] Felker, D. (2011). *Android Application Development for Dummies*. Indiana: Wiley Publishing, Inc.
- [7] “Java untuk Android” [Online]. Tersedia : <http://code.tutsplus.com/tutorials/learn-java-for-Android-development-introduction-to-java--mobile-2604> [diakses 14 Agustus 2014].
- [8] “Google Cloud Messaging” [Online]. Tersedia: <http://developer.Android.com/google/gcm/index.html> [diakses 14 Agustus 2014].

- [9] “Push Notification using Google Cloud messaging” [Online]. Tersedia:
http://Androidexample.com/Android_Push_Notifications_using_Google_Cloud_Messaging_GCM/index.php?view=article_discription&aid=119&aaid=139 [diakses 14 Agustus 2014].
- [10] “Google App Engine” [Online]. Tersedia:
<https://developers.google.com/appengine/docs/whatisgoogleappengine> [diakses 16 Agustus 2014].
- [11] “Google Cloud Endpoint” [Online]. Tersedia:
<https://developers.google.com/appengine/docs/java/endpoints/> [diakses 16 Agustus 2014].
- [12] “Java Datastore API” [Online]. Tersedia:
<https://developers.google.com/appengine/docs/java/datastore/> [diakses 16 Agustus 2014].
- [13] “Java Data Objects with AppEngine” [Online]. Tersedia:
<https://developers.google.com/appengine/docs/java/datastore/jdo/overview-dn2> [diakses 16 Agustus 2014].
- [14] “Blobstore Java API” [Online]. Tersedia:
<https://developers.google.com/appengine/docs/java/blobstore/> [diakses 16 Agustus 2014].
- [15] “Apache Velocity” [Online]. Tersedia:
<http://wiki.apache.org/velocity/FrontPage> [diakses 16 Agustus 2014].

LAMPIRAN

Absensi kehadiran dalam satu bulan adalah sebagai berikut:

Tanggal	Kehadiran	
	Nabilla Sabbaha	Ardhiansyah Baskara
23 Juni 2014	v	v
24 Juni 2014	v	v
25 Juni 2014	v	v
26 Juni 2014	v	v
27 Juni 2014	v	v
28 Juni 2014	v	v
29 Juni 2014	v	v
30 Juni 2014	v	v
1 Juli 2014	v	v
2 Juli 2014	v	v
3 Juli 2014	v	v
4 Juli 2014	v	v
5 Juli 2014	v	v
6 Juli 2014	v	v
7 Juli 2014	v	v
8 Juli 2014	v	v
9 Juli 2014	v	v
10 Juli 2014	v	v
11 Juli 2014	v	v
12 Juli 2014	v	v
13 Juli 2014	v	v
14 Juli 2014	v	v
15 Juli 2014	v	v
16 Juli 2014	v	v
17 Juli 2014	v	v
18 Juli 2014	v	v
19 Juli 2014	v	v
20 Juli 2014	v	v
21 Juli 2014	v	v
22 Juli 2014	v	v
23 Juli 2014	v	v

