BAB III
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

Dalam proses pembuatan suatu sistem mutlak dilakukan penelitian dan penganalisaan tentang sistem yang akan dibangun, berikut adalah beberapa analisis yang dilakukan untuk membangun aplikasi untuk memanfaatkan QR *code* sebagai tiket masuk dengan memperhitungkan kode koreksi kesalahan.

3.2 Analisis Masalah

Tiket merupakan suatu dokumen yang sangat penting dalam suatu acara (*event*) karena tiket menentukan apakah seseorang memiliki akses untuk memasuki acara tersebut atau tidak. Pada umumnya tiket berupa kertas yang memiliki ukuran sebesar uang kertas dengan cetakan gambar atau grafik khusus sesuai dengan keinginan penyelenggara acara. Untuk melakukan verifikasi tiket, petugas yang berjaga di pintu masuk harus memastikan bahwa setiap orang yang masuk dapat menunjukan tiket. Apabila tiket yang ditunjukan sesuai, petugas menyobekan sebagian dari tiket tersebut sebagai tanda bahwa tiket tersebut telah digunakan.

Kesederhanaan sistem tiket ini, kini banyak disalahgunakan untuk mendapatkan akses masuk ke dalam acara secara cuma-cuma. Ada yang masuk dengan menggunakan potongan tiket dan berpura-pura telah masuk sebelumnya, bahkan ada yang sampai melakukan pemalsuan tiket. Tidak adanya proses identifikasi dan validasi tiket dengan data yang dimiliki oleh penyelengara acara membuat kecurangan tetap saja terjadi.

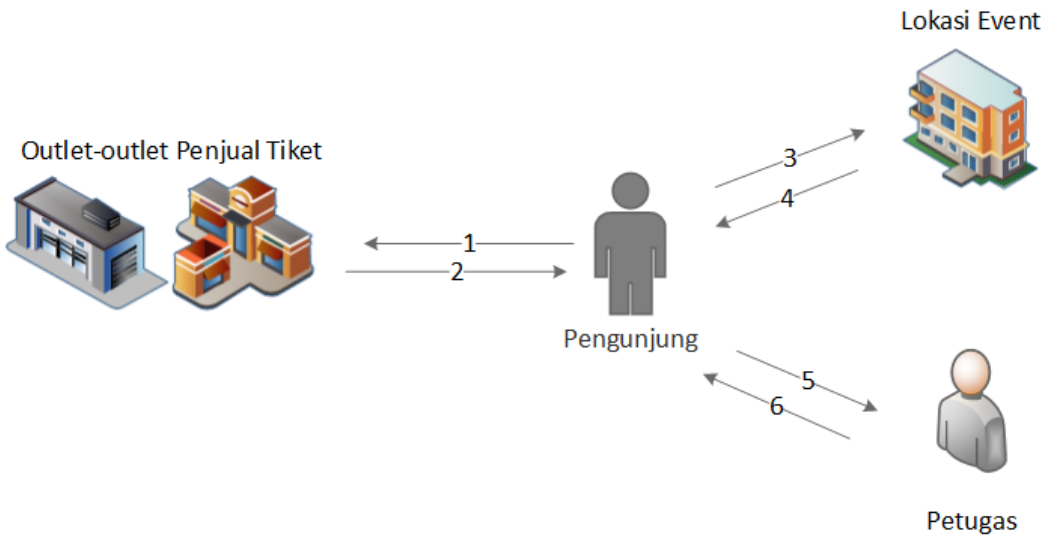
Salah satu cara penjualan tiket yang ada saat ini adalah cara penjualan *presale* atau dijual beberapa hari sebelum acara berlangsung dengan harga yang lebih murah atau pun karena banyaknya peminat *event*. Caranya, para pengunjung datang ke *outlet* atau ke tempat-tempat yang menyediakan tiket *event* dan membeli tiket disana. Akan tetapi yang didapatkan adalah nota pembelian yang untuk kemudian ditukarkan dengan tiket pada hari dilaksanakannya event. Hal ini membuat proses pembelian tiket menjadi rumit karena pengunjung harus beberapa kali datang ke tempat berbeda untuk mendapatkan tiket.

Oleh karena itu, penelitian kali ini mencoba untuk memanfaatkan QR *code* sebagai tiket masuk yang dijual secara *presale* tanpa mengelola proses pembayaran. Maka perlu dilakukan penelitian untuk mengetahui bagaimana cara memanfaatkan QR *code* sebagai tiket masuk dengan memperhitungkan tingkat koreksi kesalahan menggunakan *Reed-Solomon error correction code* untuk meningkatkan akurasi pada proses verifikasi tiket. Untuk mengetahui tingkat akurasinya tersebut, perlu dilakukan pengujian pada tiket masuk QR *code* yang dibangkitkan dengan empat tingkat koreksi kesalahan yang berbeda, yaitu *Low*, *Medium*, *Quartile* dan *High*.

3.3 Analisis Sistem Berjalan

Analisis sistem berjalan merupakan proses penguraian terhadap sistem yang telah ada atau sedang berjalan dengan tujuan untuk melakukan evaluasi yang ada, merumuskan tujuan yang ingin dicapai dengan sistem yang baru, serta menyususun rencana pembangunan sistem.

Pada dasarnya prosedur manual proses verifikasi tiket pada beberapa instansi atau pun penyelenggara acara tidak jauh berbeda. Pengunjung dapat membeli tiket di *outlet* atau mitra resmi yang penjual tiket. Kemudian pengunjung akan mendapatkan nota pembelian untuk ditukarkan pada saat *event* berlangsung. Setelah mendapatkan tiket, pengunjung melewati gerbang atau pintu masuk yang dijaga oleh petugas. Kemudian, petugas akan memeriksa tiket tersebut dan memberi tanda pada tiket tersebut dengan cara disobek untuk membuktikan bahwa tiket tersebut telah dipergunkan.

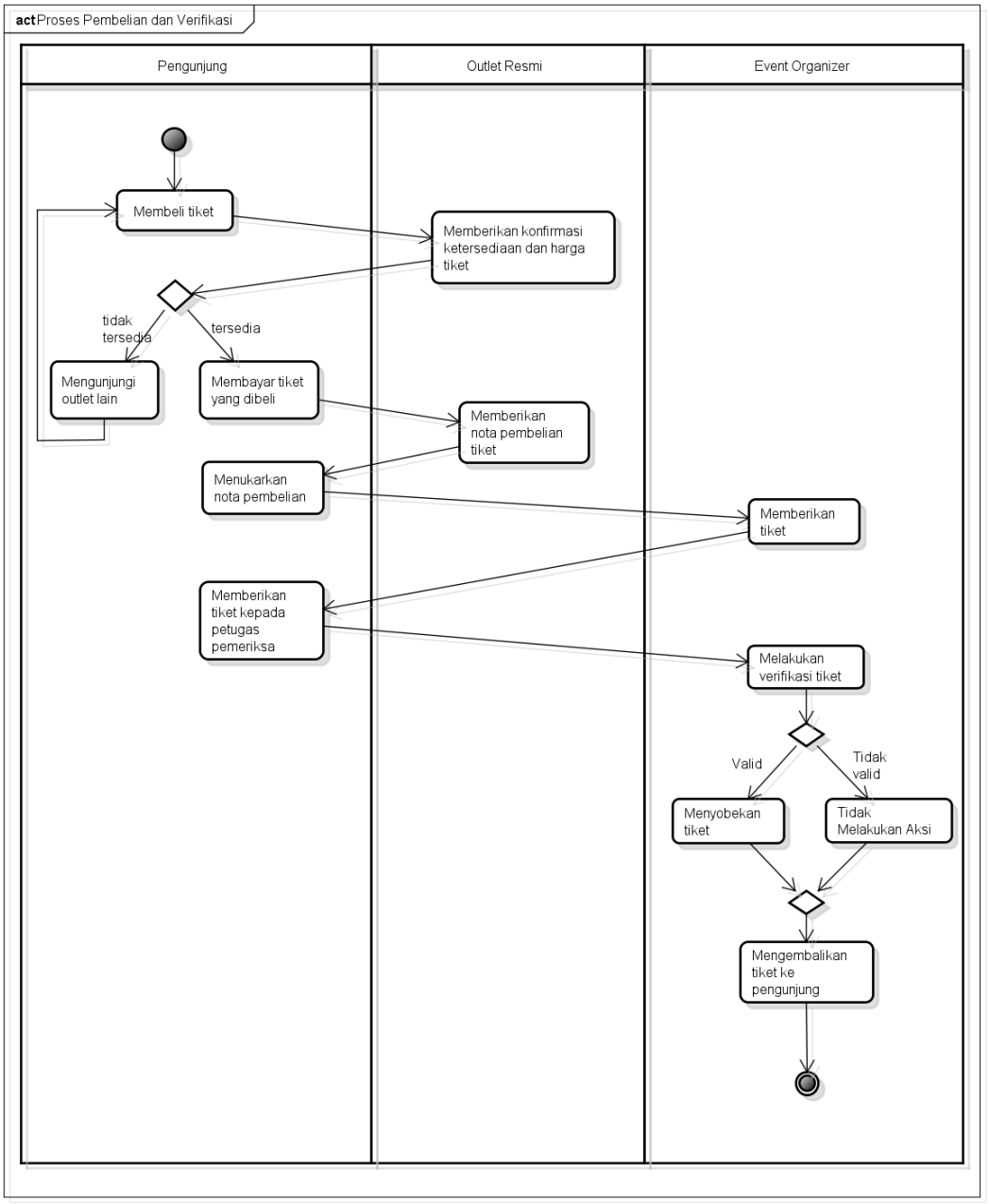


Gambar 3.1 Gambaran Umum Pembelian dan Verifikasi Tiket Presale

Keterangan:

1. Pengunjung membeli tiket dari *outlet* resmi penjual tiket.
2. *Outlet* resmi memberikan nota pembayaran sebagai bukti pembelian tiket.
3. Pengunjung menukarkan nota pembelian dengan tiket di loket yang ada pada lokasi *event* tersebut berlangsung.
4. Petugas di loket memeberikan tiket kepada pengunjung sesuai dengan nota pembelian.
5. Pengunjung memasuki pintu masuk atau gerbang dan kemudian menyerahkan tiket yang telah dibeli ke petugas penjaga pintu masuk.
6. Petugas penjaga pintu masuk memeriksa tiket tersebut. Apabila valid, petugas menyobekan tiket tersebut dan mengembalikannya pada pengunjung. Kemudian pengunjung pun dapat melewati pintu masuk.

Dari Gambar 3.1 dan penjelasan prosedur manual, dapat dibuat diagram aktifitas untuk memodelkan proses pembelian dan verifikasi tiket. Berikut diagram aktifitas untuk proses pada Gambar 3.2.



Gambar 3.2 Diagram Aktifitas Pembelian dan Verifikasi Tiket Presale

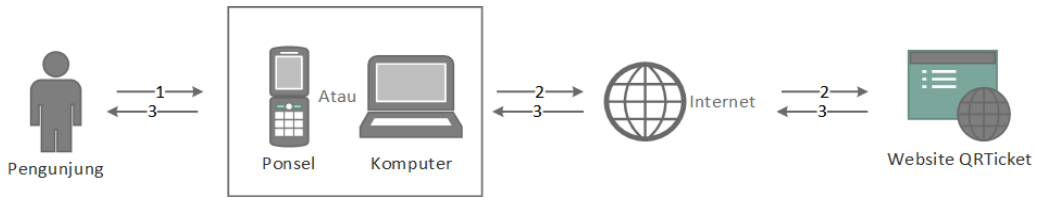
3.4 Analisis Sistem yang Dikembangkan

Analisis sistem yang dikembangkan merupakan penguraian dari suatu sistem yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

3.4.1 Deskripsi Sistem

Untuk mengganti fisik sebuah tiket menjadi QR *code*, diperlukan aplikasi yang terdiri dari *server* yang juga berfungsi sebagai *encoder* dan *scanner* yang berfungsi juga sebagai *decoder*. Pada penelitian ini, *encoder* dibangun pada *flatform website* karena tiket yang dikelola merupakan tiket yang dijual secara *presale* (dijual pada hari-hari sebelum dilaksanakannya *event*) sehingga membutuhkan sebuah *website* untuk melakukan penjualan secara *online*. Penjualan tiket dilakukan secara *online* untuk menghindari antrian yang berlebihan.

Untuk melakukan pembelian tiket, pengunjung cukup mengakes *website* penjualan tiket kemudian memilih *event* yang diinginkan. Pengunjung akan masuk ke halaman pemesanan tiket kemudian isi data diri yang ada pada form pemesanan tersebut dan pilih jumlah tiket yang akan dibeli. Kemudian, pengunjung akan mendapatkan sebuah nomor registrasi sebagai identitas pembeli pada sistem ini.



Gambar 3.3 Gambaran Umum Proses Pemesanan Tiket

Keterangan:

1) Pengunjung mengakses *website/server* menggunakan ponsel atau pun komputer melalui internet dan melakukan pemesanan tiket dengan mengisi form pemesanan dengan data yang valid.
2) Data pemesanan tiket dikirim ke server melalui *internet*.
3) Server akan memberikan kode registrasi untuk digunakan ketika mengirimkan nomor resi pembayaran.

Karena penelitian ini tidak membahas pembayaran tiket, maka diasumsikan pembayaran dilakukan dengan cara sebagai berikut.

1) Pengunjung melakukan transfer ke rekening penyelenggara acara.

2) Pengunjung mengakses *website* dan menginputkan nomor registrasi pada Form Pembayaran sebagai autentifikasi pengunjung.
3) Jika nomor registrasi valid, pengunjung mengirimkan bukti transfer atau nomor resi pembayaran melalui Form Pembayaran.
4) Setelah pengunjung mengirimkan nomor resi pembayaran, sistem akan melakukan *encoding* QR *code* dengan status "Non-aktif" dan pengunjung pun dapat mengunduh tiket dalam bentuk QR *code*. Pengunjung dapat langung menyimpannya ke dalam ponsel, karena pada proses verifikasi nanti pengunjung cukup menunjukan QR *code* yang ditampilkan dari layar ponsel pengunjung.
5) Administrator melakukan verifikasi secara manual untuk menentukan validitas dari bukti transfer yang dikirimkan pengunjung. Apabila valid, maka administrator akan melakukan *update* pada status pembayaran menjadi lunas, sehingga status tiket pun akan ikut terupdate menjadi "Aktif".

Perlu diketahui, tiket QR *code* yang ada pada sistem ini memiliki tiga jenis status yang berbeda. Hanya tiket yang berstatus "Aktif" yang dianggap valid oleh sistem ketika proses verifikasi tiket.

Tabel 3.1 Keterangan Status Tiket

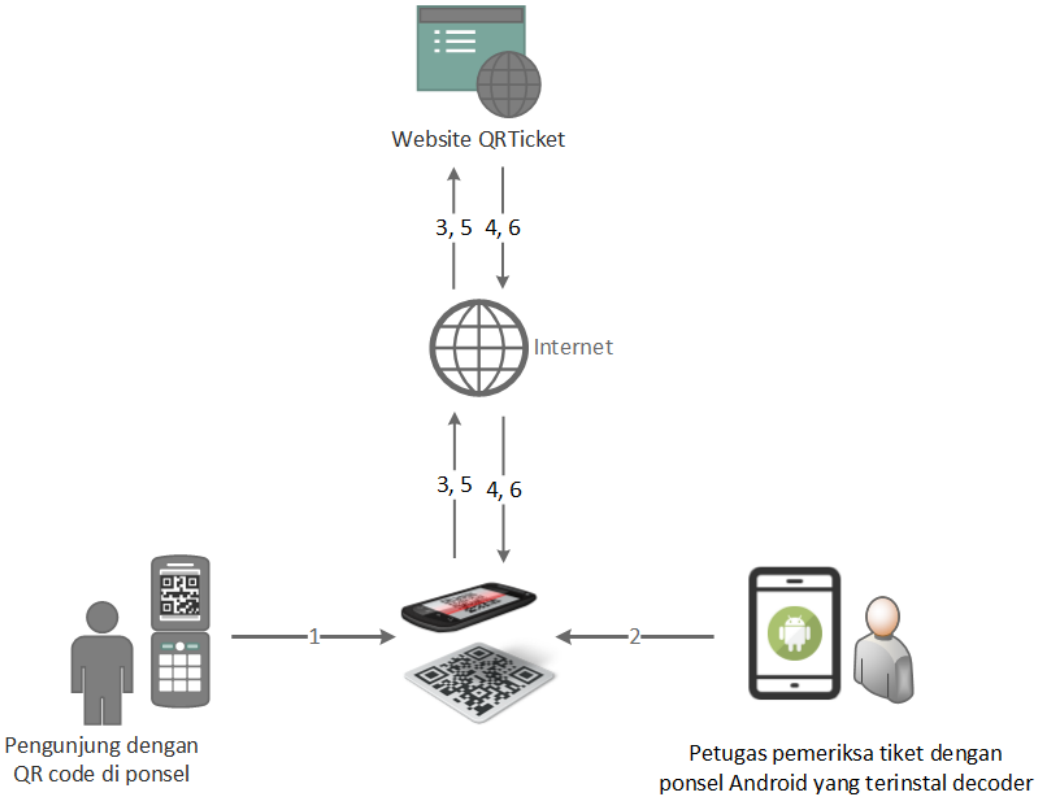
No	Status Tiket	Keterangan
1	Non-Aktif	Ketika pertama kali dibangkitkan (dimana user telah mengirimkan nomor resi pembayaran), tiket akan memiliki status "Non-aktif"
2	Aktif	Ketika administrator mengubah status pembayaran menjadi lunas (nomor resi pembayaran terbukti valid), maka status tiket yang semula "Non-aktif" akan berubah menjadi "Aktif".
3	Terverifikasi	Ketika petugas pemeriksa tiket melakukan verifikasi terhadap tiket dengan status "Aktif" maka status tiket akan berubah menjadi "Terverifikasi". Itu berarti pengunjung memiliki tiket yang valid dan

		memiliki akses untuk memasuki area *event*. Tiket dengan status "Terverifikasi" ini menandakan bahwa tiket tersebut telah digunakan sehingga dalam sistem tiket tersebut tidaklah valid

Selain proses *encoding* QR *code* dan pemesanan tiket, *website* ini juga berfungsi sebagai *server* yang menyimpan dan mengelola data acara, data pemesanan dan data tiket. Fungsional pada pengelolaan data ini berupa tambah, edit dan hapus data. Data pemesanan yang dikelola adalah status pembayaran dimana status pembayaran ini juga mempengaruhi status dari tiket yang dibeli. Seluruh pengelolaan ini dikelola oleh administrator.

Decoder pada sistem ini berupa aplikasi *scanner* yang dibangun pada platform Android. Nantinya, aplikasi *scanner* ini harus terinstal pada telepon seluler bersistem operasi Android yang memiliki kamera serta dapat terhubung dengan jaringan *internet.* Hal ini diperlukan untuk melakukan proses verifikasi secara langsung dan mencocokan data tiket dari data hasil *decoding* QR *code* dengan data tiket pada server.

Sebelum melakukan verifiaksi tiket, petugas pemeriksa tiket harus memilih *event* terlebih dahulu sesuai dengan *event* yang berlangsung dengan memasukan kode *event* yang petugas dapatkan dari administrator atau koordinator acara. Apabila tiket yang dimiliki pengunjung sesuai dengan *event* tersebut dan memiliki status "Aktif", maka proses verifikasi tiket ini akan menggati status tiket menjadi "Terverifikasi". Gambar 3.4 merupakan gambaran umum sistem pada proses verifikasi tiket.



Gambar 3.4 Gambaran Umum Proses Verifikasi Tiket

Keterangan:

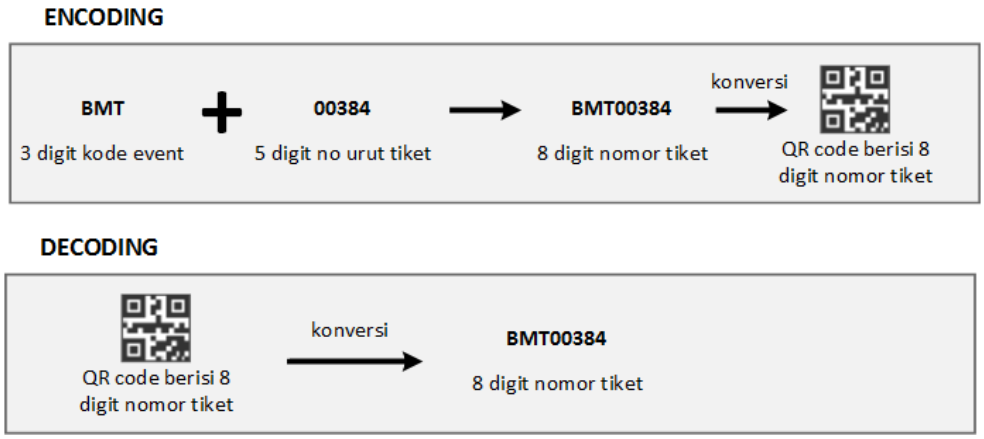
1) Pengunjung memperlihatkan tiket QR *code* yang ada pada ponselnya kepada petugas penjaga pintu masuk.
2) Petugas memilih *event* sesuai dengan *event* yang berlangsung saat itu. Kemudian petugas melukakan verifikasi tiket menggunakan aplikasi *decoder* yang terinstal pada perangkat Androidnya.
3) Jika data pada QR *code* terbaca, *decoder* akan mengirimkan data dari hasil *decoding* tersebut melalui *internet*. Kemudian server melakukan verifikasi terhadap data tersebut.
4) *Server* mengirimkan hasil validasi dari nomor tiket tersebut ke *decoder*. Apabila tiket valid (nomor tiket terdaftar, nomor *event* sesuai dengan *event* yang berlangsung dan status tiket "Aktif") maka tiket dapat diproses.
5) Petugas pemeriksa tiket menekan tombol verifikasi. Verifiaksi disini, ialah mengganti status tiket menjadi "Terverifikasi". *Decoder* akan meminta

server untuk mengganti status tiket yang "Aktif" tersebut menjadi "Terverifikasi" sebagai tanda bahwa tiket telah digunakan.

6) Jika update berhasil, *server* akan mengirimkan pesan ke *decoder* bahwa tiket telah terverifikasi.

3.4.2 Analisis Kebutuhan Data

Aplikasi yang dibangun merupakan aplikasi yang melakukan *encoding* dan *decoding* tiket menjadi QR *code*. Pada proses *encoding*, terjadi perubahan data dari nomor tiket yang bertipe string dengan panjang 8 karakter menjadi sebuah QR *code* yang berupa citra. Nomor tiket dengan jumlah 8 karakter ini, kemudian dibagi menjadi 2 bagian, dimana 3 karakter pertama merupakan kode *event* Dalam mengkonversi inputan berupa nomor tiket menjadi output berupa citra QR *code* tersebut dilakukan dalam beberapa proses yang merupakan tahapan dari proses *encoding*.



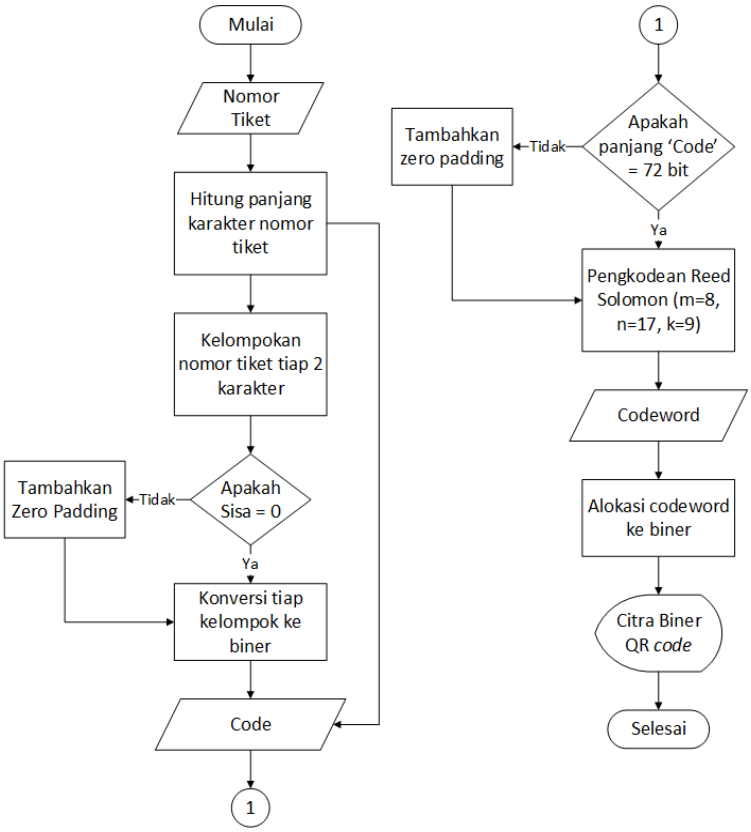
Gambar 3.5 Aliran Data pada Proses Encoding dan Decoding

Sedangkan untuk proses *decoding* dibutuhkan citra QR *code* yang akan diambil secara realtime menggunakan kamera perangkat Android untuk kemudian dari input citra tersebut diolah lebih lanjut oleh sistem untuk mendapatkan output berupa teks berisi nomor tiket. Kemudian nomor tiket dikirim ke server guna dilakukan verifikasi keaslian atau validasi dari QR *code* tersebut.

3.4.3 Encoding QR Code

Encoding QR *code* merupakan proses mengkodekan data pada QR *code.* Pada penelitian ini, data yang dikodekan berupa nomor tiket. Misalkan ambil contoh nomor tiket adalah "BMT00384" yang terdiri dari 8 digit dan QR *code* yang digunakan adalah versi 1 dengan koreksi kesalahan tingkat *High* (1-H) dimana 30% dari jumlah *codewords* data dapat dikembalikan. Ini semua berkat adanya kode koreksi kesalahan *Reed-Solomon*.

Pada proses *encoding* QR *code*, data yang akan dikodekan akan dibentuk menjadi *codewords* data. Kemudian kode koreksi kesalahan *Reed-Solomon* akan membentuk suatu *codewords* baru. *Codewords* ini akan menyimpan data yang sama dengan data yang ada pada *codewords* data yang berfungsi sebagai cadangan apabila data pada *codewords* data tidak terbaca oleh pemindai. Gambar 3.6 merupakan diagram alir dari proses *encoding* QR *code.*



Gambar 3.6 Diagram Alir Proses Encoding

Langkah-langkah yang dilakukan saat proses *encoding* data menjadi *codewords* dapat juga dituliskan dengan singkat sebagai berikut.

1) Perhitungan jumlah karakter serta pemberian indikator mode.
2) *Encoding* data dalam representasi biner.
3) Pemenuhan jumlah bit dan simbol sesuai dengan jumlah yang ditentukan
4) Menghitung *codeword* kode koreksi kesalahan Reed-Solomon.

3.4.3.1 Indikator Mode

Indikator mode dibuat dengan panjang 4 bit sebagai representasi biner seperti pada Tabel 3.2. Data yang dikodekan adalah "BMT00384", itu berarti data yang digunakan adalah alfanumerik. Maka indikator mode yang dipilih sesuai dengan Tabel 3.2 adalah **0010**.

Tabel 3.2 Keterangan Indikator Mode

Indikator	Keterangan
0001	Numeric encoding (10 bits per 3 digits)
0010	Alphanumeric encoding (11 bits per 2 characters)
0100	Byte encoding (8 bits per character)
1000	Kanji encoding (13 bits per character)
0011	Structured append (used to split a message across multiple QR symbols)
0111	Extended Channel Interpretation (select alternate character set or encoding)
0101	FNC1 in first position (see Code 128 for more information)
1001	FNC1 in second position
0000	End of message

3.4.3.2 Indikator Jumlah Karakter

Indikator jumlah karakter adalah jumlah karakter yang dapat disimpan dalam setiap mode. Seperti pada Tabel 3.3 untuk data alfanumerik, panjang representasi binernya adalah 9 bit. Panjang contoh nomor tiket adalah 8 bit, maka 8 bit direpresentasikan dalam biner adalah **000001000** sehingga bilangan biner yang sudah terbentuk adalah 0010 **000001000**.

Tabel 3.3 Jumlah Bit per Panjang Field

Encoding	Ver.1–9	10–26	27–40
Numerik	10	12	14
Alfanumerik	9	11	13
Byte	8	16	16
Kanji	8	10	12

3.4.3.3 Encoding Data dalam Representasi Biner

Dalam mode alfanumerik, setiap karakter di konversikan ke dalam nilai sesuai dengan Tabel 2.4. Data dibagi dalam kelompok-kelompok per 2 karakter dimana nilai dari karakter pertama dikali 45 hasilnya ditambah dengan nilai karakter kedua. Seperti dijelaskan pada rumus $V = 45 \times C_1 + C_2$. Kemudian nilai dari V dikodekan dalam representasi biner sepanjang 11 bit. Tabel 3.4 merupakan hasil penguraian dari nomor tiket "BMT00384".

Tabel 3.4 Encoding Data dalam Representasi Biner

"BM"	**"T0"**	**"03"**	**"84"**
45*11+22	45*29+0	45*0+3	45*8+4
517	553	3	93
01000000101	10100011001	00000000011	00101101100

Lalu lakukan *terminator* atau penambahkan biner 0000 kepada hasil representasi biner. Akan tetapi jika panjang data yang di *encode* penuh sesuai dengan versi dan tingkat koreksi kesalahan, maka tidak perlu dilakukan *terminator*.

0010 000001000 01000000101 10100011001 00000000011 00101101100 **0000**

3.4.3.4 Pemenuhan Jumlah Bit dan Jumlah Simbol

Hasil representasi data nomor tiket 0010 000001000 01000000101 10100011001 00000000011 00101101100 0000 kemudian dikelompokan setiap 8bit. Sehingga hasil representasi tersebut menjadi:

00100000 01000010 00000101 10100011 00100000 00001100 10110110 00000

Jika terdapat kelompok dengan panjang data kurang dari 8 bit, maka tambahkan 0 sampai menjadi 8 bit. Penambahan "0" inilah yang dinamakan *zero padding.* Penambahan *zero padding* tersebut diperlukan karena jumlah simbol yang menjadi masukan Reed Solomon harus tetap yaitu 8 bit disebut dengan *codeowrds*. Setelah dilakukan *zero padding*, maka kelompok atau deret *codewords* tersebut menjadi:

00100000 01000010 00000101 10100011 00100000 00001100 10110110 00000**000**

Jika jumlah *codewords* kurang dari kapasitas simbol, maka tambahkan byte "11101100" dan "00010001" secara bergantian hingga panjang bit memenuhi kapasitas maksimum yang telah ditentukan. Byte tersebut masing-masing ekuivalent dengan 236 dan 17. Kedua byte tersebut secara khusus diperlukan oleh spesifikasi QR *code* untuk ditambahkan jika bit string terlalu pendek pada tahap ini.

Untuk kapasitas maksimum QR *code* versi 1 dengan tingkat koreksi kesalahan *High*, dapat dilihat pada Table 2.6 dan didapat bahwa QR *code* 1-H memiliki kapasitas maksimum 9 *codewords.* Jika dikalikan dengan 8 bit maka representasi biner tersebut harus berjumlah 72 bit.

00100000	01000010	00000101	10100011	00100000	00001100	10110110	00000000
1	2	3	4	5	6	7	8

Setelah dihitung, jumlah *codewords* pada representasi biner berjumlah 8 *codewords* dengan jumlah bit sebesar 64 bit sehingga perlu dilakukan

penambahan byte "11101100". Maka kapasitas dari representasi biner tersebut telah berjumlah 9 *codewords*.

00100000	01000010	00000101	10100011	00100000
1	2	3	4	5
00001100	10110110	00000000	**11101100**	
6	7	8	**9**	

Kemudian *codewords* tersebut dikonversi kembali kedalam bilangan desimal sehingga diperoleh deret desimal **32 66 5 163 32 12 182 0 236.** Deret desimal tersebut nantinya akan menjadi masukan dalam proses pengkodean Reed-Solomon.

3.4.3.5 Menghitung Codewords Kode Koreksi Kesalahan

Dalam QR *code* digunakan kode koreksi kesalahan dari Redd-Solomon. Pertama hasil *encoding* data dari bagian 3.4.3.4 yaitu **32 66 5 163 32 12 182 0 236** dipisah sesuai ketentuan dari blok RS pada Tabel 2.6. Dapat dilihat pada kolom "Banyak blok pada Group 1", jika menunjukan angka 1 maka tidak perlu dilakukan pemisahan data. Untuk 1-H adalah 1, sehingga data pun tidak perlu dipisah. Kemudian pilih rumus g(x) dari Tabel 3.5 berdasarkan jumlah *Error-Correction codewords* per blok dari Tabel 2.6 sesuai dengan versi QR *code* dan tingkat koreksi kesalahan.

Tabel 3.5 Generator Polynomial Kode Koreksi Kesalahan

Jumlah EC codewords per blok	**Generator Polinomial g(x)**
7	x7+α87x6+α229x5+α146x4 +α149x3+α238x2+α102x+α21
10	x10+α251x9+α67x8+α46x7+α61x6+α118x5+α70x4+α64x3+α94 x2+α32x+α45
13	x13+α74x12+α152x11+α176x10+α100x9+α86x8+α100x7+α106 x6+α104x5+α130x4+α218x3+α206x2+α140x +α78
16	x16+α120x15+α104x14+α107x13+α109x12+α102x11+α161x10

	+α76x9+α3x8+α91x7+α191x6+α147x5+α169x4 +α182x3+α194x2+α225x +α120
17	x17+α43x16+α139x15+α206x14+α78x13+α43x12+α239x11+α123x10+α206x9+α214x8+α147x7+α24x6+α99x5 +α150x4+α39x3+α243x2 +α163x+α136
18	x18+α215x17+α234x16+α158x15+α94x14+α184x13+α97x12+α118x11+α170x10+α79x9+α187x8+α152x7+α148x6+α252x5+α179x4+α5x3+α98x2+α96x+α153
22	x22+α210x21+α171x20+α247x19+α242x18+α93x17+α230x16+α14x15+α109x14+α221x13+α53x12+α200x11+α74x10+α8x9+α172x8+α98x7+α80x6+α219x5+α134x4 +α160x3+α105x2+α165x +α231
28	x28+α168x27+α223x26+α200x25+α104x24+α224x23+α234x22 +α108x21+α180x20+α110x19+α190x18+α195x17+α147x16+α205x15+α27x14+α232x13+α201x12+α21x11+α43x10+α245x9+α87x8+α42x7+α195x6+α212x5+α119x4+α242x3+α37x2+α9x +α123

Untuk 1-H jumlah EC (*error correction*) *codewords* adalah 17 sehingga rumus g(x) yang digunakan adalah sebagai berikut.

$$g(x)=x^{17}+\alpha^{43}x^{16}+\alpha^{139}x^{15}+\alpha^{206}x^{14}+\alpha^{78}x^{13}+\alpha^{43}x^{12}$$
$$+\alpha^{239}x^{11}+\alpha^{123}x^{10}+\alpha^{206}x^{9}+\alpha^{214}x^{8}+\alpha^{147}x^{7}+\alpha^{24}x^{6}$$
$$+\alpha^{99}x^{5}+\alpha^{150}x^{4}+\alpha^{39}x^{3}+\alpha^{243}x^{2}+\alpha^{163}x+\alpha^{136}$$

Pada rumus tersebut, α adalah elemen primitif pada *Galois Field* (2^8) atau disingkat GF(2^8). Beberapa fitur dari GF(2^8) itu sendiri adalah:

1) Mendukung 4 operasi aritmatika
2) α255 = 1
3) Dapat mengkonversi eksponensial dalam α ke integer (atau sebaliknya) menggunakan Tabel 3.6.

Tabel 3.6 Tabel Korespondensi dan Indeks Integer dari GF(2^8)α

Exponent of α	Integer	Exponent of α	Integer	Exponent of α	Integer	Exponent of α	Integer
0	1	**64**	95	**128**	133	**192**	130
1	2	**65**	190	**129**	23	**193**	25
2	4	**66**	97	**130**	46	**194**	50
3	8	**67**	194	**131**	92	**195**	100
4	16	**68**	153	**132**	184	**196**	200
5	32	**69**	47	**133**	109	**197**	141
6	64	**70**	94	**134**	218	**198**	7
7	128	**71**	188	**135**	169	**199**	14
8	29	**72**	101	**136**	79	**200**	28
9	58	**73**	202	**137**	158	**201**	56
10	116	**74**	137	**138**	33	**202**	112
11	232	**75**	15	**139**	66	**203**	224
12	205	**76**	30	**140**	132	**204**	221
13	135	**77**	60	**141**	21	**205**	167
14	19	**78**	120	**142**	42	**206**	83
15	38	**79**	240	**143**	84	**207**	166
16	76	**80**	253	**144**	168	**208**	81
17	152	**81**	231	**145**	77	**209**	162
18	45	**82**	211	**146**	154	**210**	89
19	90	**83**	187	**147**	41	**211**	178
20	180	**84**	107	**148**	82	**212**	121
21	117	**85**	214	**149**	164	**213**	242
22	234	**86**	177	**150**	85	**214**	249
23	201	**87**	127	**151**	170	**215**	239
24	143	**88**	254	**152**	73	**216**	195
25	3	**89**	225	**153**	146	**217**	155
26	6	**90**	223	**154**	57	**218**	43

27	12	**91**	163	**155**	114	**219**	86
28	24	**92**	91	**156**	228	**220**	172
29	48	**93**	182	**157**	213	**221**	69
30	96	**94**	113	**158**	183	**222**	138
31	192	**95**	226	**159**	115	**223**	9
32	157	**96**	217	**160**	230	**224**	18
33	39	**97**	175	**161**	209	**225**	36
34	78	**98**	67	**162**	191	**226**	72
35	156	**99**	134	**163**	99	**227**	144
36	37	**100**	17	**164**	198	**228**	61
37	74	**101**	34	**165**	145	**229**	122
38	148	**102**	68	**166**	63	**230**	244
39	53	**103**	136	**167**	126	**231**	245
40	106	**104**	13	**168**	252	**232**	247
41	212	**105**	26	**169**	229	**233**	243
42	181	**106**	52	**170**	215	**234**	251
43	119	**107**	104	**171**	179	**235**	235
44	238	**108**	208	**172**	123	**236**	203
45	193	**109**	189	**173**	246	**237**	139
46	159	**110**	103	**174**	241	**238**	11
47	35	**111**	206	**175**	255	**239**	22
48	70	**112**	129	**176**	227	**240**	44
49	140	**113**	31	**177**	219	**241**	88
50	5	**114**	62	**178**	171	**242**	176
51	10	**115**	124	**179**	75	**243**	125
52	20	**116**	248	**180**	150	**244**	250
53	40	**117**	237	**181**	49	**245**	233
54	80	**118**	199	**182**	98	**246**	207
55	160	**119**	147	**183**	196	**247**	131
56	93	**120**	59	**184**	149	**248**	27

57	186	**121**	118	**185**	55	**249**	54
58	105	**122**	236	**186**	110	**250**	108
59	210	**123**	197	**187**	220	**251**	216
60	185	**124**	151	**188**	165	**252**	173
61	111	**125**	51	**189**	87	**253**	71
62	222	**126**	102	**190**	174	**254**	142
63	161	**127**	204	**191**	65	**255**	1

1) Langkah Divisi Polinomial

Langkah pertama untuk divisi ini adalah mempersiapkan polinomial pesan untuk divisi. Polinom pesan lengkap adalah:

$32x^8 + 66x^7 + 5x^6 + 163x^5 + 32x^4 + 12x^3 + 182x^2 + 0x^1 + 23$

Untuk memastikan bahwa eksponen dari variabel pertama tidak menjadi terlalu kecil selama divisi, kalikan polinomial pesan dengan xn dimana n adalah jumlah *codewords* koreksi kesalahan yang diperlukan. Dalam hal ini n adalah 17, untuk 17 *codewords* koreksi kesalahan, kalikan polinomial pesan oleh x17, maka akan mendapatkan hasil:

$32x^{25} + 66x^{24} + 5x^{23} + 163x^{22} + 32x^{21} + 12x^{20} + 182x^{19} + 0x^{18} + 236x^{17}$

Istilah utama dari generator polinomial juga harus memiliki eksponen yang sama, jadi kalikan dengan x^8 untuk mendapatkan

$\alpha^0x^{25} + \alpha^{43}x^{24} + \alpha^{139}x^{23}\ \alpha^{206}x^{22} + \alpha^{78}x^{21}\ \alpha^{43}x^{20} + \alpha^{239}x^{19} + \alpha^{123}x^{18} + \alpha^{206}x^{17} + \alpha^{214}x^{16} + \alpha^{147}x^{15}\ \alpha^{24}x^{14} + \alpha^{99}x^{13}\ \alpha^{150}x^{12} + \alpha^{39}x^{11}\ \alpha^{243}x^{10} + \alpha^{163}x^9 + \alpha^{136}x^8$

Sekarang sudah mungkin untuk melakukan langkah-langkah pembagian berulang. Jumlah langkah di divisi harus sama dengan jumlah variabel dalam polinomial pesan. Dalam hal ini, divisi ini akan mengambil 9 langkah untuk menyelesaikan. Hal ini akan menghasilkan sisa yang memiliki 17 variabel. Istilah-istilah ini akan menjadi 17 *codewords* koreksi kesalahan yang diperlukan.

2) Langkah 1a: Kalikan Generator Polinomial dengan Variabel Pertama pada Polinomial Pesan

Langkah pertama adalah untuk memperbanyak generator polinomial dengan variabel pertama dari polinomial pesan. Dalam studi kasus ini, variabel pertama adalah $32x^{25}$. Karena notasi α membuatnya lebih mudah untuk melakukan perkalian, dianjurkan untuk mengkonversi $32x^{25}$ notasi α. Menurut Tabel 3.3, untuk nilai integer 32, eksponen α adalah 5 . Oleh karena $32 = \alpha^5$. Kalikan generator polinomial oleh α^5 :

$(\alpha^5 * \alpha^0) x^{25} + (\alpha^5 * \alpha^{43}) x^{24} + (\alpha^5 * \alpha^{139}) x^{23} + (\alpha^5 * \alpha^{206}) x^{22} + (\alpha^5 * \alpha^{78}) x^{21} + (\alpha^5 * \alpha^{43}) x^{20} + (\alpha^5 * \alpha^{239}) x^{19} + (\alpha^5 * \alpha^{123}) x^{18} + (\alpha^5 * \alpha^{206}) x^{17} + (\alpha^5 * \alpha^{214}) x^{16} + (\alpha^5 * \alpha^{147}) x^{15} + (\alpha^5 * \alpha^{24}) x^{14} + (\alpha^5 * \alpha^{99}) x^{13} + (\alpha^5 * \alpha^{150}) x^{12} + (\alpha^5 * \alpha^{39}) x^{11} + (\alpha^5 * \alpha^{243}) x^{10} + (\alpha^5 * \alpha^{163}) x^9 + (\alpha^5 * \alpha^{136}) x^8$

Eksponen dari α ditambahkan bersama-sama. Hasilnya adalah:

$\alpha^5x^{25} + \alpha^{48}x^{24} + \alpha^{144}x^{23} + \alpha^{21}1x^{22} + \alpha^{83}x^{21} + \alpha^{48}x^{20} + \alpha^{244}x^{19} + \alpha^{128}x^{18} + \alpha^{211}x^{17} + \alpha^{219}x^{16} + \alpha^{152}x^{15} + \alpha^{29}x^{14} + \alpha^{104}x^{13} + \alpha^{155}x^{12} + \alpha^{44}x^{11}\ \alpha^{248}x^{10} + \alpha^{168}x^9 + \alpha^{141}x^8$

Sekarang , konversikan kembali ke notasi integer:

$32x^{25} + 70x^{24} + 168x^{23} + 178x^{22} + 187x^{21} + 70x^{20} + 250x^{19} + 133x^{18} + 178x^{17} + 86x^{16} + 73x^{15} + 48x^{14} + 13x^{13} + 114x^{12} + 238x^{11} + 27x^{10} + 252x^9 + 21x^8$

3) Langkah 1b: XOR hasil dengan polinomial pesan

Karena ini adalah langkah divisi pertama, lakukan operasi XOR hasil dari 1a dengan polinomial pesan.

$(32 \oplus 32) x^{25} + (66 \oplus 70) x^{24} + (5 \oplus 168) x^{23} + (163 \oplus 178) x^{22} + (32 \oplus 187) x^{21} + (12 \oplus 70)\ ^{x20} + (182 \oplus 250) x^{19} + (0 \oplus 133) x^{18} + (236 \oplus 178) x^{17} + (0 \oplus 86) x^{16} + (0 \oplus 73) x^{15} + (0 \oplus 48) x^{14} + (0 \oplus 13) x^{13} + (0 \oplus 114) x^{12} + (0 \oplus 238) x^{11} + (0 \oplus 27) x^{10} + (0 \oplus 252) x^9 + (0 \oplus 21) x^8$

Hasilnya adalah :

$0x^{25} + 4x^{24} + 17x^{22} + 173x^{23} + 155x^{21} + 74x^{20} + 76x^{19} + 133x^{18} + 94x^{17} + 86x^{16} + 73x^{15} + 48x^{14} + 13x^{13} + 114x^{12} + 238x^{11} + 27x^{10} + 252x^{9} + 21x^{8}$

Buang variabel depan 0 untuk mendapatkan:

$4x^{24} + 17x^{22} + 173x^{23} + 155x^{21} + 74x^{20} + 76x^{19} + 133x^{18} + 94x^{17} + 86x^{16} + 73x^{15} + 48x^{14} + 13x^{13} + 114x^{12} + 238x^{11} + 27x^{10} + 252x^{9} + 21x^{8}$

4) Langkah 2 Lakukan Iterasi Hingga Akhir

Setelah mendapatkan hasil XOR pada langkah 1b, lakukan kembali iterasi seperti pada langkah 1a dan 1b hingga mendapatkan hasil:

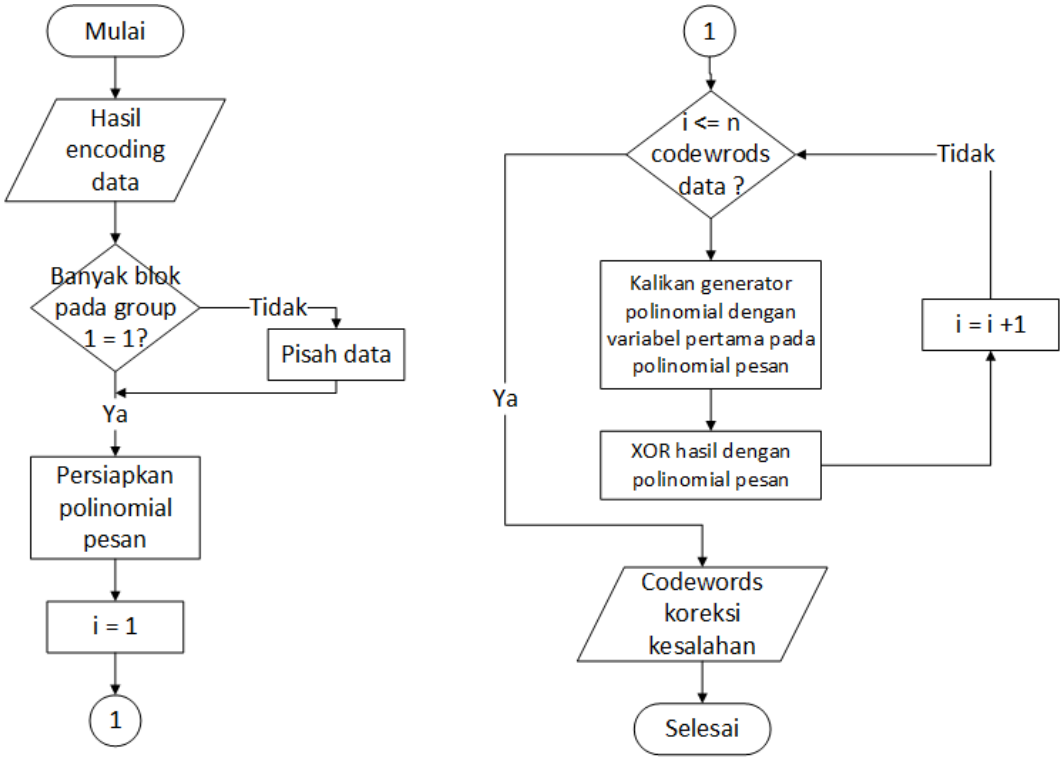
$76x^{16} + 27x^{15} + 190x^{14} + 249x^{13} + 106x^{12} + 237x^{11} + 203x^{10} + 70x^{9} + 90x^{8} + 167x^{7} + 150x^{6} + 71x^{5} + 99x^{4} + 75x^{3} + 32x^{2} + 230x^{1} + 106$

5) Menggunakan variabel sisanya sebagai *codewords* koreksi kesalahan

Divisi ini telah dilakukan 9 kali, yang merupakan jumlah variabel dalam polinomial pesan. Ini berarti bahwa iterasi selesai dan ketentuan polinomial di atas adalah *codewords* koreksi kesalahan yang digunakan untuk pesan polinomial asli:

76 27 190 249 106 237 203 70 90 167 150 71 99 75 32 230 106

Seluruh proses menghitung *codewors* koreksi kesalahan dapat juga dilihat dalam bentuk diagram alir. Gambar 3.7 merupakan gambar diagram alir tersebut.

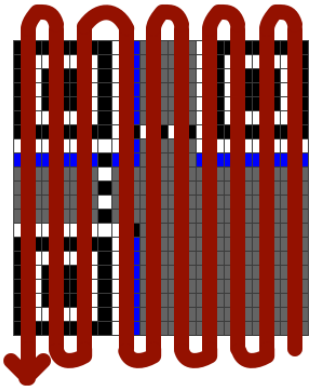


Gambar 3.7 Diagram Alir Mengitung Codeword Koreksi Kesalahan

3.4.3.6 Alokasi Codewords ke Biner

Pada QR *code*, 1 modul berarti 1 bit. Data hasil di bagian sebelumnya dikodekan ke representasi biner, kemudian data tersebut dialokasikan. Selain itu, ketika nomor blok RS adalah 2 atau lebih tinggi, maka data harus dialokasikan dalam *interleaved*. Dalam contoh data, untuk 1-H (QR *code* versi 1 dengan tingkat koreksi kesalahan H) memiliki 1 RS blok, data tidak perlu dialokasikan dalam *interleave*. Ada beberapa aturan untuk alokasi data, diantaranya:

1) Pengalokasian memiliki koordinat, i menunjukan baris dan j menunjukan kolom. Pojok kiri atas, memiliki koordinat (0,0). Untuk QR *code* versi 1 ini, memiliki modul dari (0,0) hingga (20,20).
2) Awal modul adalah pojok kanan bawah. Pada QR *code* 1-H ini, awal modul adalah (20,20).
3) Setiap *codewords* terdiri dari 8 bit, sehingga pengalokasian pun dikelompokan menjadi 8 modul dengan ukuran modul 2x4.
4) Pola pengalokasian data dimulai dari bawah keatas kemudian dari atas kebawah, terus berlanjut hingga akhir, seperti pada Gambar 3.8 berikut.



Gambar 3.8 Pola Alokasi Data QR Code

5) Ketika alokasi data sedang mengarah keatas, maka data pun akan dialokasikan secara zig-zag ke arah atas. Mulai dari (20,20), kemudian berlanjut ke (19,20), (19,19), (20,19) hingga (20,17) kemudian dilanjutkan ke *codeword* berikutnya.
6) Ketika alokasi data sedang mengarah kebawah, maka data pun akan dialokasikan ke arah bawah, tetap dengan pola zigzag.

Setelah didapatkan desimal *codewords* data dan *codewords* koreksi kesalahan dari bagian sebelumnya maka tambahkan *codewords* koreksi kesalahan ke *codewords* data sehingga didapatkan:

32 66 5 163 32 12 182 0 236 76 27 190 249 106 237 203 70 90 167 150 71 99 75 32 230 106

Kemudian konversikan kembali *codewords* ke biner dan lakukan operasi XOR. Kemudian deretan bit-bit *codewords* data tersebut dialokasikan seperti yang telah dijelaskan sebelumnya untuk mengisi modul-modul yang ada pada QR *code*. Posisi *codewords* koreksi kesalahan akan berada tepat setelah *codewords* data. Setelah dilakukan pengalokasian *codewords* pada setiap modul, maka didapat hasil citra QR *code* untuk nomor tiket “BMT00384” seperti pada Gambar 3.9.

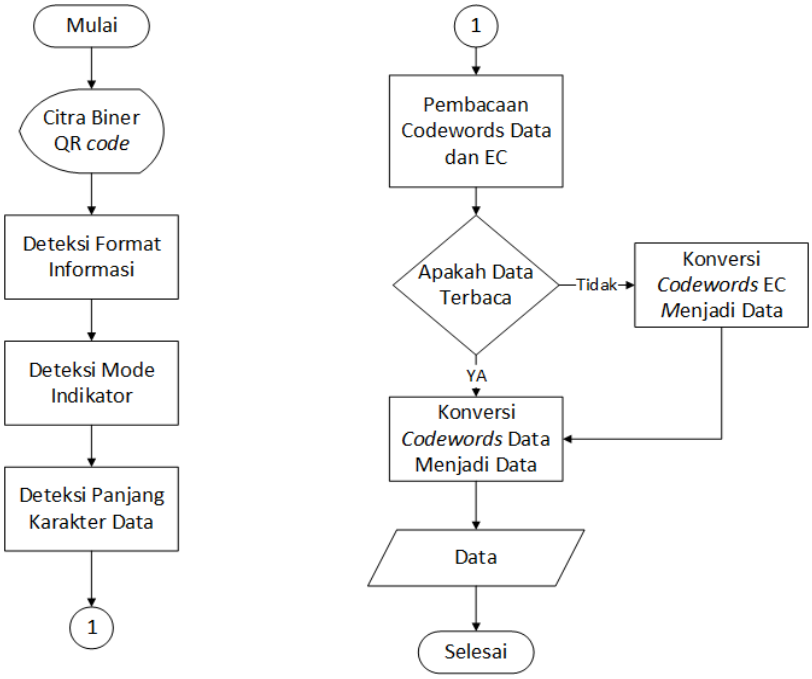


Gambar 3.9 QR Code Hasil Encoding Data "BMT00384"

3.4.4 Decoding QR Code

Sistem *decoder* yang dibangun berbasis *mobile* untuk mempermudah mobilitas pemeriksa tiket dalam memeriksa tiketnya. *Decoder* ini akan memanfaatkan kamera yang ada pada perangkat *mobile* untuk menangkap QR *code*. Kemudian QR *code* tersebut akan di *decode* untuk mengetahui isi data nomor tiket yang ada pada QR *code.*

Proses *decoding* akan memanfaatkan *library* dari ZXing sehingga *decoder* akan melakukan pemanggilan *class* yang telah disertakan dari ZXing. Apabila QR *code* berhasil di *decode* maka, *decoder* akan menampilkan data hasil *decoding* tersebut dan kemudian mengirimkan data tersebut ke server untuk kemudian dilakukan verifikasi tiket. Gambar 3.10 merupakan diagram alir dari proses *decoding*.



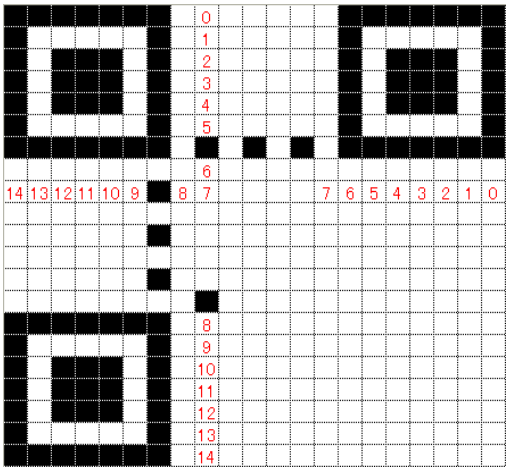
Gambar 3.10 Diagram Alir Proses Decoding QR Code

Langkah-langkah yang dilakukan pada proses *decoding* QR *code* menjadi data dapat juga dituliskan dengan singkat sebgai berikut.

1) Deteksi format informasi, mode indikator dan panjang karakter data.
2) Pembacaan isi pada QR *code* yang berupa *codewords.*
3) Konversi kembali *codewords* data menjadi data.

3.4.4.1 Deteksi Format Informasi dan Mode Indikator

Ketika memulai untuk melalukan *decoding* QR *code*, langkah pertama yang dilakukan adalah mendeteksi format informasi. Format informasi terdiri dari 5 bit. 2 bit pertama adalah untuk tingkat koreksi kesalahan. Sedangkan 3 bit berikutnya adalah untuk *masking number.* Pada QR *code* terdapat dua salinan format informasi. Namun hanya dibutuhkan 1 satu saja untuk proses *decoding.* Format terletak pada posisi seperti pada Gambar 3.11.



Gambar 3.11 Format Informasi pada QR Code

Kemudian dilakukan deteksi mode indikator yang berada pada pojok kanan bawah. Mode indikator ini terdiri dari 4 bit dengan posisi 2x2. Dapat dilihat pada tanda merah di Gambar 3.12.



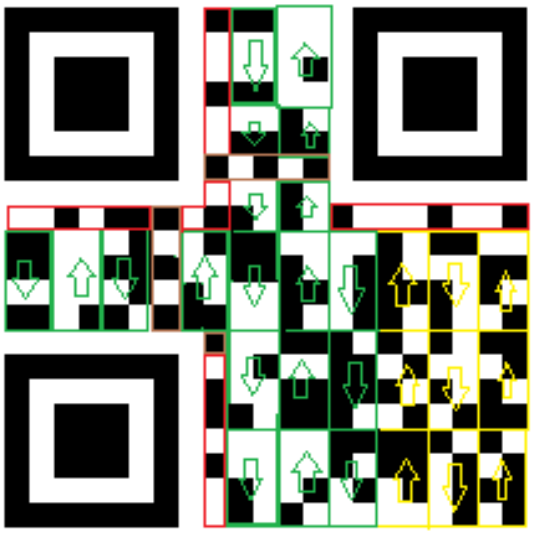
Gambar 3.12 Mode Indikator dan Panjang Karakter pada QR Code

Untuk tanda biru pada Gambar 3.12 merupakan panjang data pada QR *code.* Cara pembacaan bagi ketiganya, cukup dilihat pada warna QR *code.* Hitam atau gelap menandakan 1 sedangkan putih atau terang menunjukan 0. Kemudian lihat 5 bit format informasi dari Gambar 3.12 maka didapat **"00101"**. Sebenarnya angka tersebut adalah hasil XOR dari **10101** sebelum di cetak pada QR *code.* Maka dari hasil 00101^10101 = **10000**. Dua digit pertama (**10**) merupakan tingkat koreksi kesalahan yang digunakan pada QR *code.* Apabila dicocokan dengan tabel 2.5

maka dapat diketahui bahwa tingkat koreksi kesalahan yang digunakan adalah level H.

3.4.4.2 Pembacaan Codewords pada isi QR code

Isi atau data pada QR *code* terletak pada bagian tengah atau pada kotak kosong yang ditunjukan oleh Gambar 3.11. Untuk lebih jelasnya, posisi untuk *codewords* data dan *codewords* koreksi kesalahan dapat dilihat pada Gambar 3.13 dimana *codewords* untuk data BMT00384 terletak pada blok berwarna kuning yang berjumlah 9 blok. Untuk blok berwarna hijau, blok tersebut berisi kode koreksi kesalahan yang berjumlah 17 blok.



Gambar 3.13 Penempatan Pesan dan Simbol QR Code Berisi "BMT00384"

3.4.4.3 Konversi Codewords Menjadi Data

Seperti sudah dibahas sebelumnya, setiap *codewords* terdiri dari 8 bit. Ambil saja *codewords* pertama dari QR *code* pada Gambar 3.13. jika direpresentasikan pada bilangan biner, maka akan menghasilkan **10111001** kemudian lakukan opersai XOR dengan **10101001** maka akan menghasilkan **00010000**. Kemudian konversikan bilangan tersebut ke desimal sehingga didapat integer 32. Apabila bilangan tersebut disamakan dengan codewords desimal yang di encode, maka hasilnya sama-sama 32 begitu seterusnya proses pembacaan dan konversi *codewords* menjadi data lebih tepatnya data nomor tiket BMT00384.

3.4.5 Pembacaan Tiket

Proses pembacaan tiket merupakan proses validasi untuk menentukan apakah QR *code* yang diberikan pengunjung merupakan tiket yang valid atau bukan. Sebelum melakukan validasi tiket, petugas pemeriksa tiket harus memasukan kode *event* terlebih dahulu untuk memastikan bahwa tiket yang diverifikasi merupakan tiket *event* yang bersesuai dengan *event* yang berlangsung.

Setelah *event* yang dipilih valid, maka dilakukan *decoding* pada tiket QR *code* tersebut. Setelah itu, didapatkan hasil *decoding* yang berupa data yang seharusnya adalah nomor tiket. Kemudian data tersebut dikirimkan ke server melaui internet. Setelah itu server akan mencocokan data tersebut dengan nomor tiket yang ada pada *database.* Selain nomor tiket yang harus sama dengan yang ada di *database,* terdapat kriteria tiket yang valid, didasarkan akan 2 hal, yaitu:

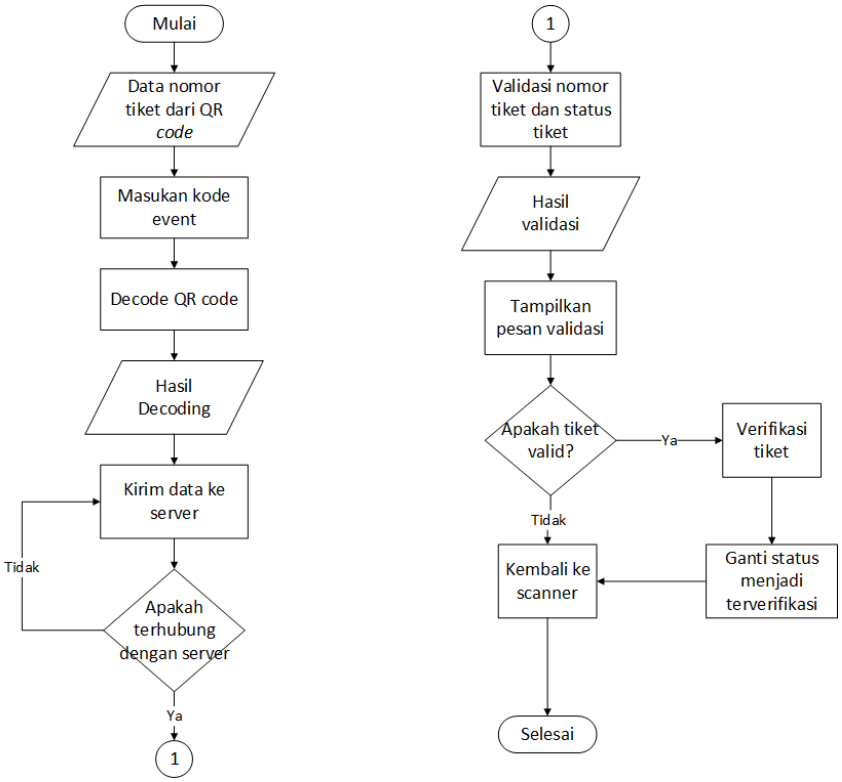
1) Kode *Event*

   Kode *event* harus lah valid dan sesuai dengan *event* yang sedanga berlangsung

2) Status Tiket

   Ada tiga jenis status tiket, yaitu Non-aktif, Aktif dan terverifikasi. Hanya tiket yang memiliki status Aktif lah yang dinyatakan valid oleh sistem.

Setelah *server* melakukan validasi dengan *database,* maka server akan mengirimkan hasil validasi tersebut. Jika data tersebut merupakan nomor tiket yang valid, maka petugas pemeriksa tiket dapat melakukan verifikasi tiket. Perintah verifikasi tersebut dikirimkan ke *server* dan kemudian *server* akan mengganti status tiket dari yang asalnya Aktif menjadi Terverifikasi sehingga tiket tersebut kini tidak lagi valid. Gambar 3.14 berikut adalah alur proses pembacaan tiket.



Gambar 3.14 Diagram Alur Proses Pembacaan Tiket

3.5 Analisis Kebutuhan Non-Fungsional

Ananlisis kebutuhan non-fungsional merupakan penguraian kebutuhan-kebutuhan non fungsional (batasan sistem) menjadi kebutuhan yang lebih detail berdasarkan kebutuhan yang ada. Spesifikasi non-fungsional melibatkan beberapa kebutuhan, perangkat lunak, perangkat keras dan pengguna.

3.5.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak merupakan penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi perangkat lunak dan berhubungan dengan proses pembangunan perangkat lunak.

Kebutuhan perangkat lunak:

1) Sistem operasi (Windows Vista atau lebih baru)
2) *Web Server* Wamp versi 2 sebagai *web server*
3) *Code Editor* berupa Notepad++
4) Android SDK
5) PHP QR Code dan ZXing *library*

3.5.2 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat lunak merupakan penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi perangkat keras dan berhubungan dengan proses pembangunan perangkat lunak.

Kebutuhan Perangkat Keras untuk Server:

1) Processor minimal Pentium 4 3.0 GHz socket LGA 775 (disesuaikan dengan sistem operasi yang digunakan).
2) RAM 512 MB atau lebih (disesuaikan dengan sistem operasi yang digunakan).
3) Hardisk minimal 20 GB (untuk Windows 7) disesuaikan dengan sistem operasi yang digunakan.
4) *Network Interface Card* untuk jaringan komputer.

Kebutuhan Perangkat Keras untuk *Decoder*:

1) Ponsel atau tablet bersistem operasi Android.
2) Memiliki kamera.
3) Memiliki konektifitas nirkabel (Wi-Fi).

3.5.3 Analisis Kebutuhan Perangkat Pikir

Analisis kebutuhan perangkat lunak merupakan penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi pengguna dan berhubungan dengan perangkat lunak.

Tabel 3.7 Kebutuhan Perangkat Pikir

Pengguna Sistem	Hak Akses	Keterampilan yang Harus Dimiliki	Jenis Pelatihan yang Akan Diberikan
Administrator	Mengelola data *event*, data tiket	Dapat menggunakan internet dan mengerti tentang andministrasi	Mengenai tata cara penggunaan CMS pada *website*
Pemeriksa Tiket	Melakukan verifikasi tiket	Dapat menggunakan	Mengenai penggunaan *decoder*

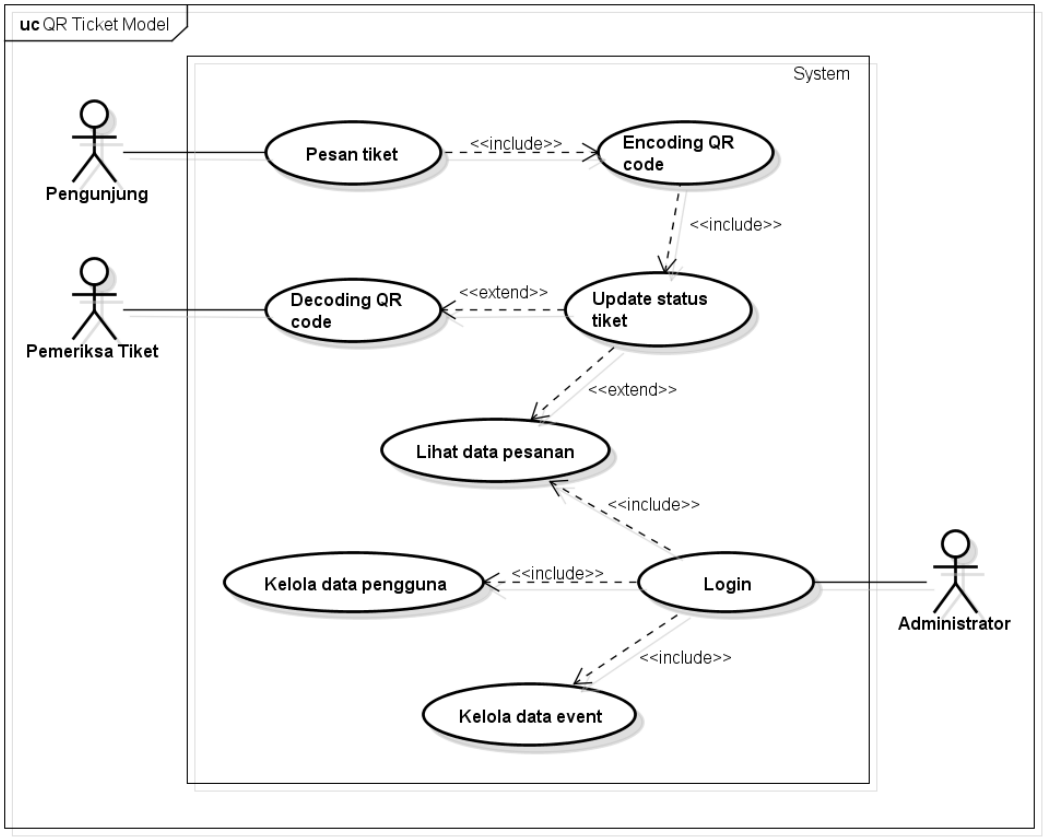
		perangkat bersistem opersai Android	pada perangkat Android
Pengunjung	Melakukan pemesanan tiket	Dapat menggunakan internet	-

3.6 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan penguraian dari kebutuhan yang berkaitan dengan fungsi atau proses transformasi yang harus mampu dikerjakan oleh perangkat lunak seperti fitur, karakteristik, dan batasan lainnya yang berhubungan dengan sistem.

3.6.1 Diagram Use Case

Didalam sistem terdapat tiga pengguna yaitu Administrator, Pemeriksa Tiket dan Pengunjung. Administrator merupakan pengguna yang memiliki hak askses untuk mengelola data *event* dan data tiket. Lalu pemeriksa tiket merupakan pengguna yang bertugas untuk melalukan verifikasi tiket. Sedangkan pengunjung merupakan pengunjung event yang melakukan pemesanan tiket melalui *website*. Tugas ketiga pengguna tersebut sebagai aktor dapat dilihat pada diagram *Use Case* pada Gambar 3.15.



Gambar 3.15 Diagram Use Case

3.1.6.1 Definisi Aktor

Definisi aktor merupakan penjelasan dari aktor yang ada pada diagram *use case*. Berikut merupakan definisi aktor dari *use case* pada Gambar 3.15.

Tabel 3.8 Definisi aktor pada use case

No	Aktor	Definisi
1.	Administrator	Petugas *Event Organizer* yang memiliki kewenangan untuk mengelola server tiket.
2.	Pemeriksa Tiket	Petugas *Event Organizer* yang memiliki kewenangan untuk memeriksa tiket dan melakukan verifikasi tiket.
3.	Pengunjung	Orang yang mengunjungi *event* dan melakukan pemesanan dan registrasi nomor resi untuk mendapatkan tiket.

3.1.6.2 Definisi Use Case

Definisi *use case* merupakan penjelasan dari *use case* yang ada pada diagram *use case*. Berikut merupakan definisi aktor dari *use case* pada Gambar 3.15.

Tabel 3.9 Definisi use case pada diagram use case

No	Use Case	Definisi
1.	Login	Proses login sebgai autentifikasi kewenangan administrator
2.	Kelola Data Pengguna	Memiliki fungsional menampilkan data pengguna dan untuk mengelola data pengguna seperti tambah data, *update* data dan hapus data.
3.	Kelola Data Event	Memiliki fungsional menampilkan data *event* dan untuk mengelola data *event* seperti tambah data, *update* data dan hapus data.
4.	Lihat Data Pesanan	Memiliki fungsional untuk menampilkan data pesanan tiket dan melihat nomor resi yang dikirimkan pengunjung. Kemudian bila nomor resi valid, administrator dapat menggati status pembayaran.
5.	Update Status Tiket	Memliki fungsional untuk mengganti status tiket berdasarkan status pembayaran dan ketika verifikasi tiket di pintu masuk.
6.	Decode QR code	Merupakan fungsional untuk melakukan *decoding* QR *code* dan melakukan verifikasi keaslian tiket.
7.	Pesan Tiket	Memiliki fungsional untuk melayani pemesanan tiket yang dilakukan pengunjung dan *submit* nomor resi pembayaran.
8.	Encode QR code	Merupakan fungsional untuk membangkitkan QR *code* berdasarkan nomor tiket ketika pengunjung men-*submit* nomor resi.

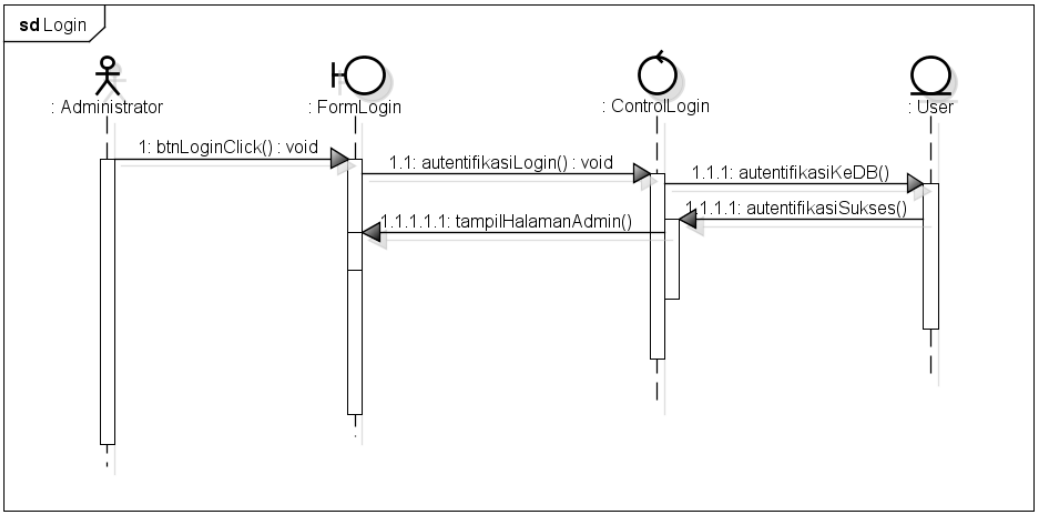
3.1.6.2.1 Login

Proses login merupakan proses yang mengawali semua sistem administrator. Login ini melakukan autentifikasi terhadap pengguna untuk menentukan bahwa pengguna yang mengakses sistem adalah administrator.

Tabel 3.10 Skenario Use Case Login

Use Case	**Login**	
Related Requirement	Sistem menyediakan form login	
Goal in Context	Administrator berhasil login kedalam sistem	
Preconditions	Sistem menampilkan halaman login	
Successful End Conditions	Login berhasil	
Failed End Conditions	Login gagal	
Primary Actors	Administrator	
Secondary Actors		
Trigger	Sistem mengakses data pengguna pada database	
Included Class	ControlLogin	
Main Flow	Step	Action
	1	Administrator menginputkan username
	2	Administrator menginputkan password
	3	Admin menekan tombol login
	4	Sistem menampilkan pesan login berhasil
	5	Sistem menampilkan halaman administrator
Extensions	Step	Branching Action
	4.1	Sistem menampilkan pesan login gagal
	5.1	Sistem tidak menampilkan halaman administrator

Proses autentifikasi mencocokan username dan password yang diinputkan administrator dengan database yang ada pada sistem. Jika username dan password cocok, tandanya autentifikasi berhasil.



Gambar 3.16 Sequence Login

3.1.6.2.2 Kelola Data Pengguna

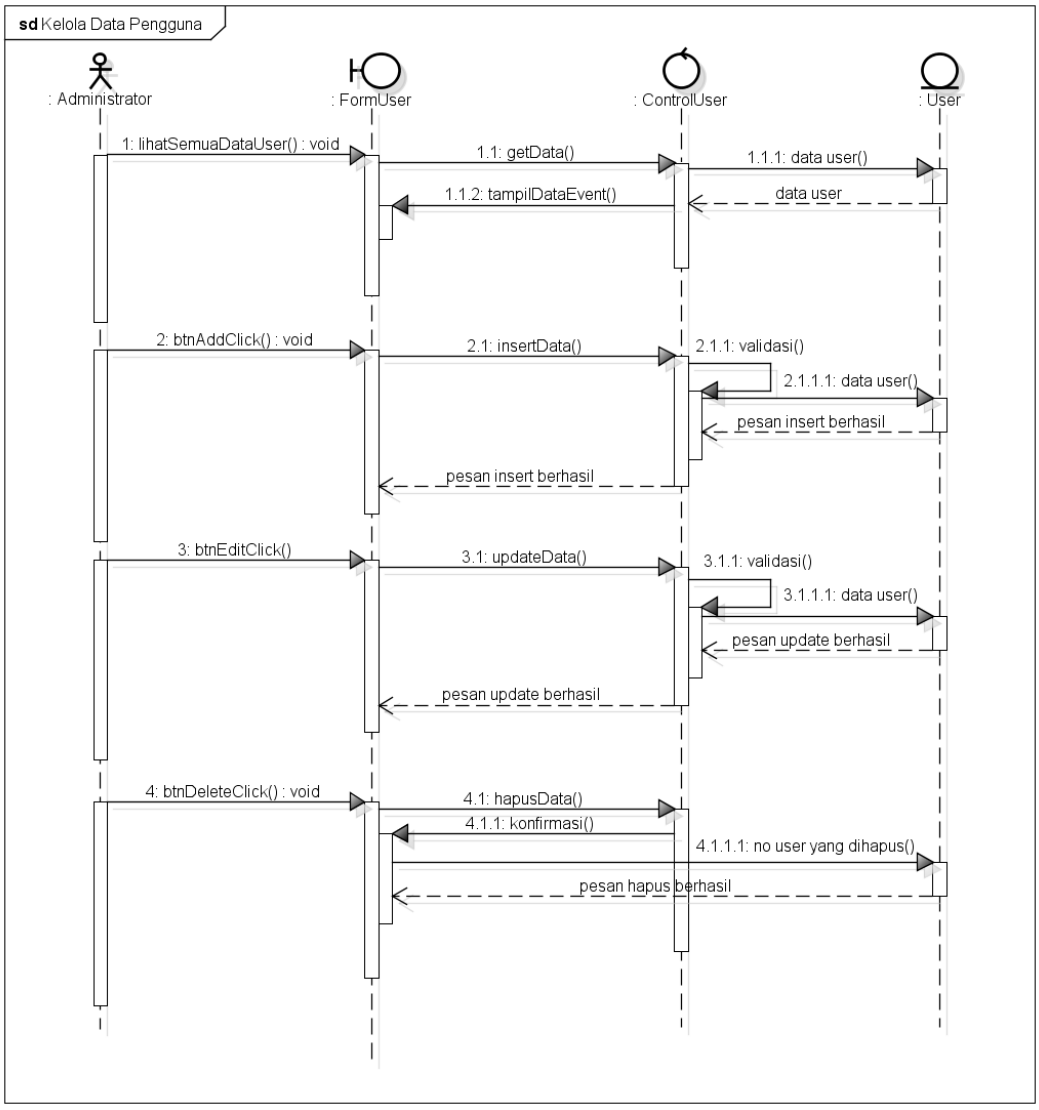
Kelola data pengguna merupakan proses yang memiliki fungsional menampilkan data pengguna dan mengelolanya. Pengelolaan data pengguna diantaranya seperti tambah data, update data dan hapus data.

Tabel 3.11 Skenario Use Case Kelola Data Pengguna

Use Case	**Kelola Data Pengguna**	
Related Requirement	Sistem menyediakan form kelola pengguna	
Goal in Context	Administrator dapat mengelola data pengguna	
Preconditions	Sistem menampilkan halaman pengguna	
Successful End Conditions	Kelola data berhasil	
Failed End Conditions	Kelola data gagal	
Primary Actors	Administrator	
Secondary Actors		
Trigger	Sistem mengakses data pengguna pada database	
Included Class	ControlUser	
Main Flow	Step	Action
	1	Sistem menampilkan data pengguna
	2	Administrator menambah data pengguna baru
	3	Administrator menginputkan nama dan data pengguna lainnya
	4	Administrator menekan tombol tambah
	5	Sistem menampilkan pesan berhasil

Extensions	Step	Branching Action
	2.1	Administrator mengedit data pengguna
	2.2	Administrator menghapus data pengguna
	5.1	Sistem menampilkan pesan gagal

Terdapat empat jenis proses yang ada pada fungsional kelola data pengguna. Diantaranya tampil data pengguna, tambah data pengguna, edit data pengguna dan hapus data pengguna.



Gambar 3.17 Sequence Kelola Data Pengguna

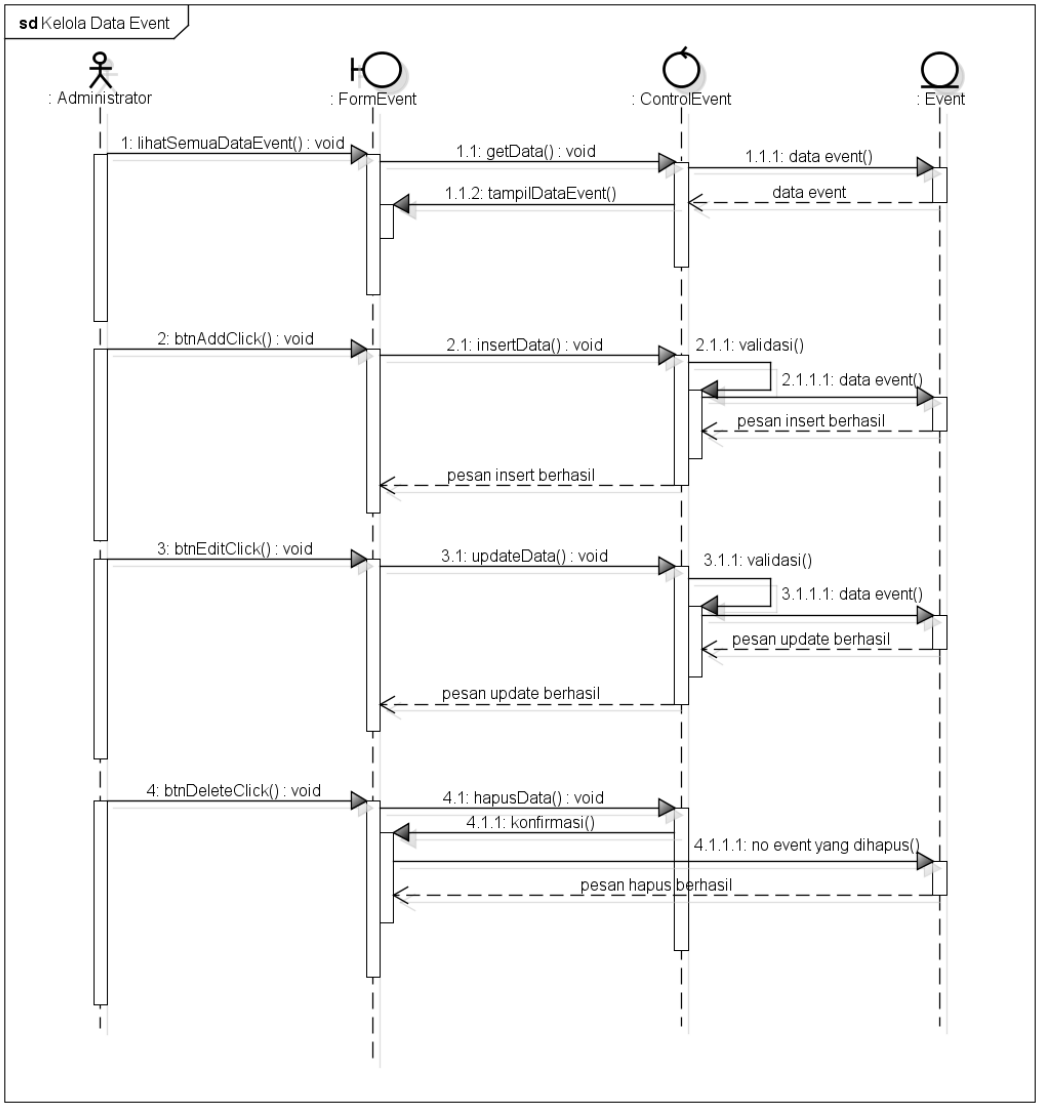
3.1.6.2.3 Kelola Data Event

Kelola data *event* merupakan proses yang memiliki fungsional menampilkan data *event* dan mengelolanya. Pengelolaan data *event* diantatanya seperti tambah data, update data dan hapus data.

Tabel 3.12 Skenario Use Case Kelola Data Event

Use Case	**Kelola Data Event**	
Related Requirement	Sistem menyediakan form kelola *event*	
Goal in Context	Administrator dapat mengelola data *event*	
Preconditions	Sistem menampilkan halaman *event*	
Successful End Conditions	Kelola data berhasil	
Failed End Conditions	Kelola data gagal	
Primary Actors	Administrator	
Secondary Actors		
Trigger	Sistem mengakses data *event* pada database	
Included Class	ControlEvent	
Main Flow	Step	Action
	1	Sistem menampilkan data *event*
	2	Administrator menambah data *event* baru
	3	Administrator menginputkan nama dan data event lainnya
	4	Administrator menekan tombol tambah
	5	Sistem menampilkan pesan berhasil
Extensions	Step	Branching Action
	2.1	Administrator mengedit data *event*
	2.2	Administrator menghapus data *event*
	5.1	Sistem menampilkan pesan gagal

Terdapat empat jenis proses yang ada pada fungsional kelola data *event*. Diantaranya tampil data *event*, tambah data *event*, edit data *event* dan hapus data *event*.



Gambar 3.18 Sequence Kelola Data Event

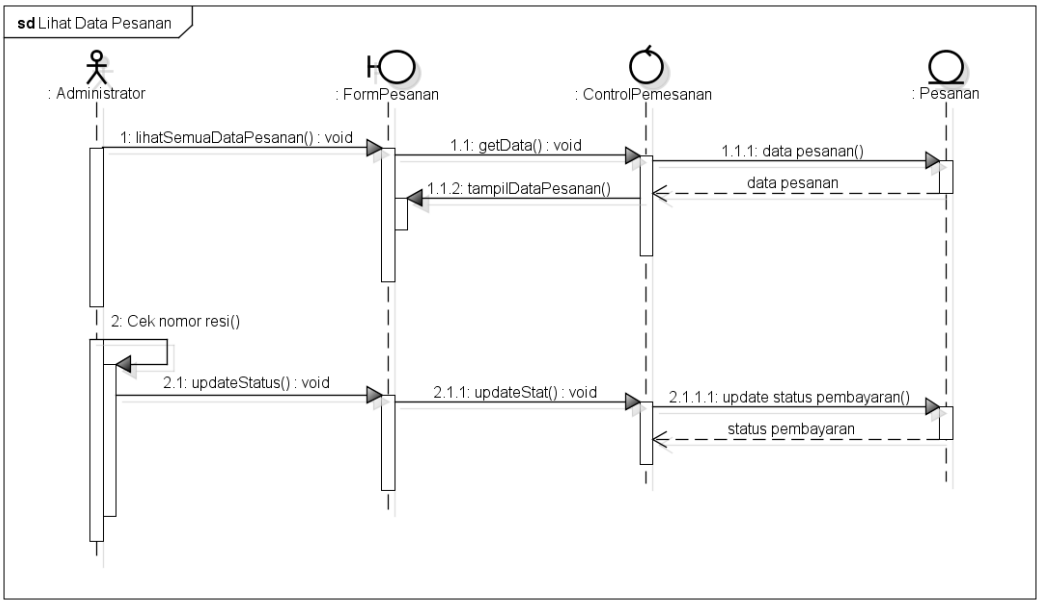
3.1.6.2.4 Lihat Data Pesanan

Lihat Data Pesanan merupakan fungsional yang berfugnsi untuk menampilkan data pesanan pada administrator untuk kemudian dilakukan pengecekan terhadap nomor resi pembayaran.

Tabel 3.13 Skenario Use Case Lihat Data Pesanan

Use Case	**Lihat Data Pesanan**	
Related Requirement	Sistem menyediakan form data pesanan	
Goal in Context	Administrator dapat mengelola data pesanan	
Preconditions	Sistem menampilkan halaman pesanan	
Successful End Conditions	Sistem menampilkan data pesanan	
Failed End Conditions	Sistem tidak menampilkan data pesanan	
Primary Actors	Administrator	
Secondary Actors		
Trigger	Sistem mengakses data pesanan pada database	
Included Class	ControlPemesanan	
Main Flow	Step	Action
	1	Sistem menampilkan data pesanan
	2	Administrator memverifikasi nomor resi
Extensions	Step	Branching Action
	2.1	Administrator melakukan *update* status pembayaran

Pada fungsional lihat data, administrator dapat juga melakukan update status pembayaran apabila nomor resi pembayaran valid.



Gambar 3.19 Sequence Lihat Data Pesanan

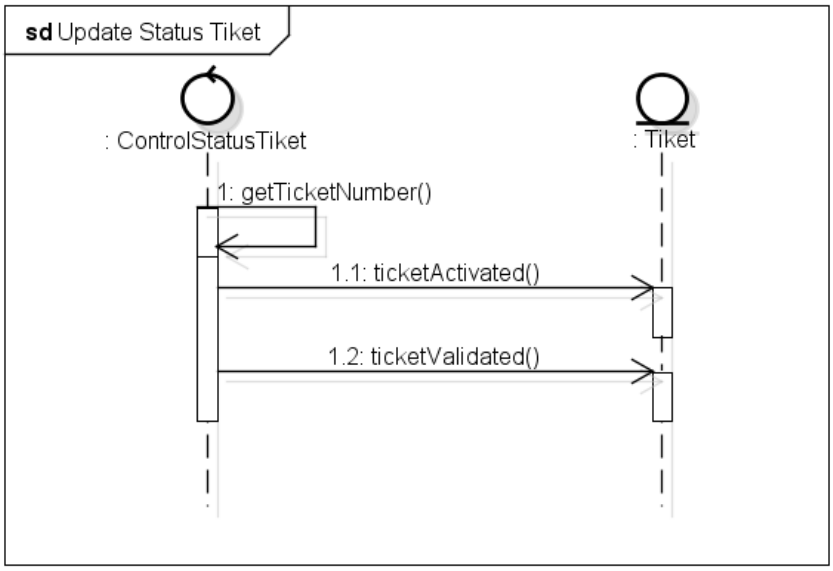
3.1.6.2.5 Update Status Tiket

Update status tiket merupakan proses yang memliki fungsional untuk mengganti status tiket. Status tiket ini berdasarkan pada status pembayaran dan ketika verifikasi tiket di pintu masuk.

Tabel 3.14 Skenario Update Status Tiket

Use Case	**Update Status Tiket**	
Related Requirement	Sistem menyediakan form data pesanan	
Goal in Context	Status tiket dapat di-*update*	
Preconditions	Administrator mengganti status pembayaran atau petugas melakukan *decode* QR code	
Successful End Conditions	Status tiket ter-*update*	
Failed End Conditions	Status tiket tidak ter-*update*	
Primary Actors	Sistem	
Secondary Actors		
Trigger	Sistem mengakses data tiket pada database	
Included Class	ControlStatusTiket	
Main Flow	Step	Action
	1	Sistem mencari nomor tiket
	2	Sistem mengganti status tiket
Extensions	Step	Branching Action
	2.1	Sistem tidak mengganti status tiket

Update status tiket akan berjalan ketika ada perubahan pada status pembayaran oleh administrator. Tiket yang tervalidasi pada proses *decoding* pun akan di *update* statusnya menjadi terverifikasi.



Gambar 3.20 Sequence Update Status Tiket

3.1.6.2.6 Decode QR Code

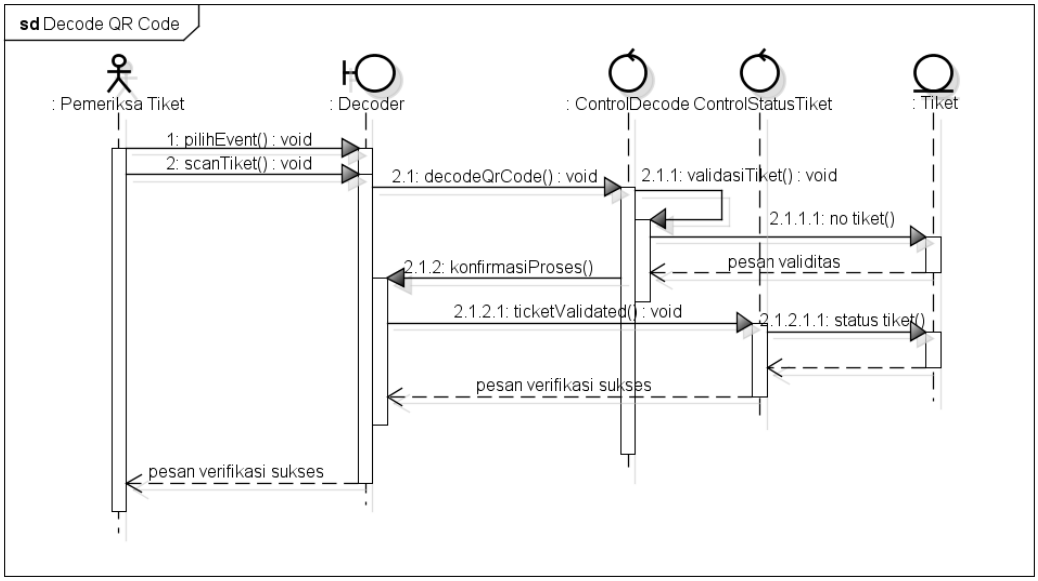
Deocode QR *code* merupakan fungsional untuk melakukan *decoding* QR *code.* Hasil *decoding* QR *code* tersebut akan diverifikasi untuk mengetahui keaslian tiket.

Tabel 3.15 Skenario Decode QR Code

Use Case	**Decode QR Code**	
Related Requirement	*Decoder* terhubung dengan *server*	
Goal in Context	Tiket dapat diverifikasi keasliannya	
Preconditions	Pemeriksa tiket menjalankan aplikasi *scanner*	
Successful End Conditions	Verifikasi tiket berhasil	
Failed End Conditions	Verifikasi tiket gagal	
Primary Actors	Pemeriksa Tiket	
Secondary Actors		
Trigger	Sistem mengakses ZXing *barcode scanner*	
Included Class	ControlDecode	
Main Flow	Step	Action
	1	Pemeriksa Tiket memilih jenis *event*

	2	Pemeriksa Tiket memindai tiket
	3	Sistem men-*decode* QR *code*
	4	Sistem melakukan verifikasi tiket
	5	Sistem mendapatkan nomor tiket untuk diperbaharui statusnya
Extensions	Step	Branching Action
	4.1	Sistem meminta konfirmasi untuk memproses tiket
	4.2	Pemeriksa Tiket memberikan konfirmasi untuk memproses tiket
	5.1	Sistem tidak mendapatkan nomor tiket

Pada proses *decoding*, sistem akan meminta konfirmasi jika tiket valid. Apabila pemeriksa tiket memberikan konfirmasi dengan memjawab ya, maka sistem akan melakukan *update* pada status tiket.



Gambar 3.21 Sequence Decode QR Code

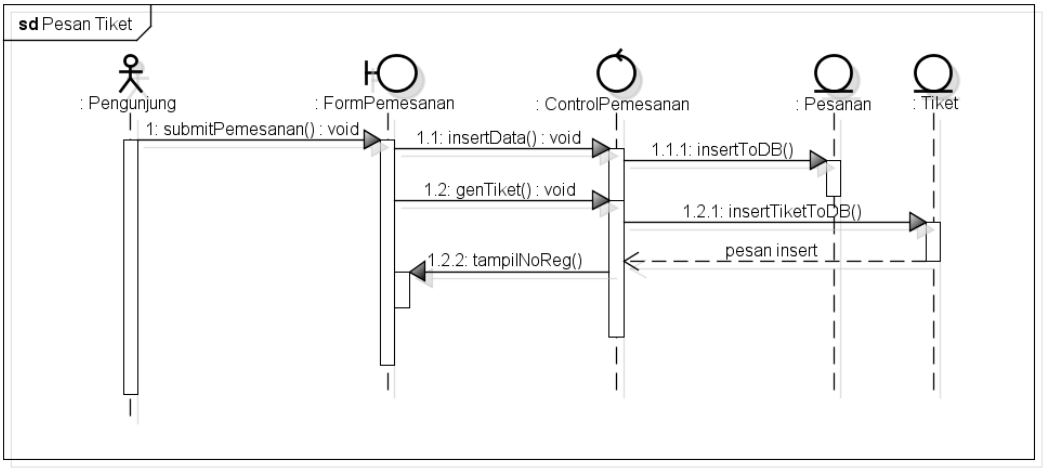
3.1.6.2.7 Pesan Tiket

Pesan tiket merupakan fungsional dimana dalam terdapat proses untuk melayani pemesanan tiket yang dilakukan pengunjung. Selain itu terdapat juga fungsional untuk mengirimkan nomor resi pembayaran.

Tabel 3.16 Skenario Pesan Tiket

Use Case	**Pemesanan Tiket**	
Related Requirement	Sistem menyediakan halaman pemesanan tiket	
Goal in Context	Pengunjung dapat memesan tiket	
Preconditions	Pengunjung mengakses halaman pemesanan tiket	
Successful End Conditions	Pemesanan tiket berhasil	
Failed End Conditions	Pemesanan tiket gagal	
Primary Actors	Pengunjung	
Secondary Actors		
Trigger	Sistem mengakses data pada database	
Included Class	ControlPemesanan	
Main Flow	Step	Action
	1	Pengunjung menginputkan data pemesanan
	2	Sistem menyimpan data pemesanan
	3	Sistem menampilkan nomor registasi pengunjung
Extensions	Step	Branching Action
	1.1	Pengunjung tidak menginputkan data pemesanan
	3.1	Sistem tidak menampilkan nomor resi

Pada pesan tiket, pengunjung mengisikan data pemesanan kemudian menekan tombol *submit*. Setelah itu sistem akan melakukan *insert* data kedalam *database*.



Gambar 3.22 Sequence Pesan Tiket

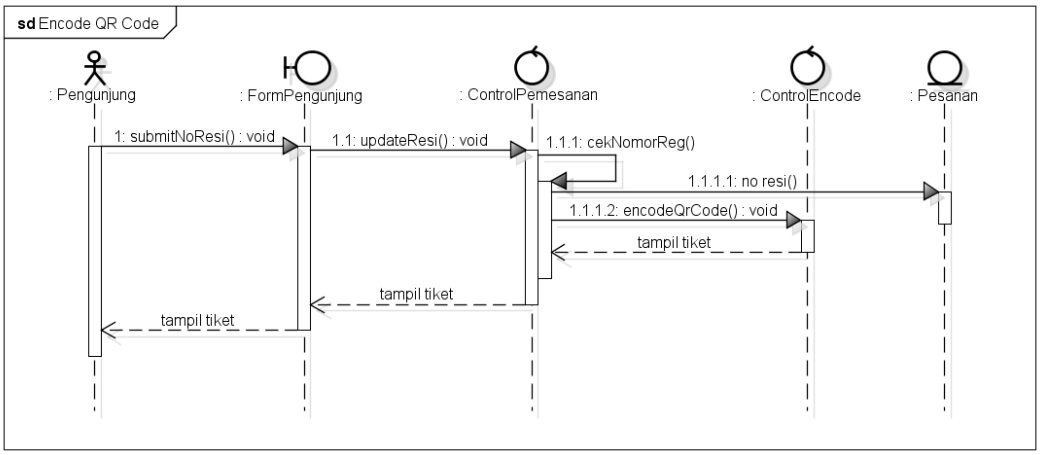
3.1.6.2.8 Encode QR Code

Encode QR *code* merupakan fungsional untuk membangkitkan QR *code* berdasarkan nomor tiket ketika pengunjung mengirimkan nomor resi. Ketika tombol submit ditekan maka sistem akan melalukan *encoding* QR *code.*

Tabel 3.17 Skenario Encode QR Code

Use Case	**Encode QR Code**	
Related Requirement	Sistem menyediakan form input nomor resi	
Goal in Context	Pengunjung mendapatkan tiket	
Preconditions	Pengunjung mengakses form input nomor resi	
Successful End Conditions	*Encode* tiket berhasil	
Failed End Conditions	*Encode* tiket gagal	
Primary Actors	Pengunjung	
Secondary Actors	Sistem	
Trigger	Sistem melakukan insert pada tabel tiket	
Included Class	ControlEncode	
Main Flow	Step	Action
	1	Pengunjung menginputkan nomor registrasi dan nomor resi
	2	Sistem membangkitkan QR *code*
	3	Sistem menampilkan QR *code*
Extensions	Step	Branching Action
	2.1	Sistem tidak membangkitkan QR *code*
	3.1	Sistem tidak menampilkan QR *code*

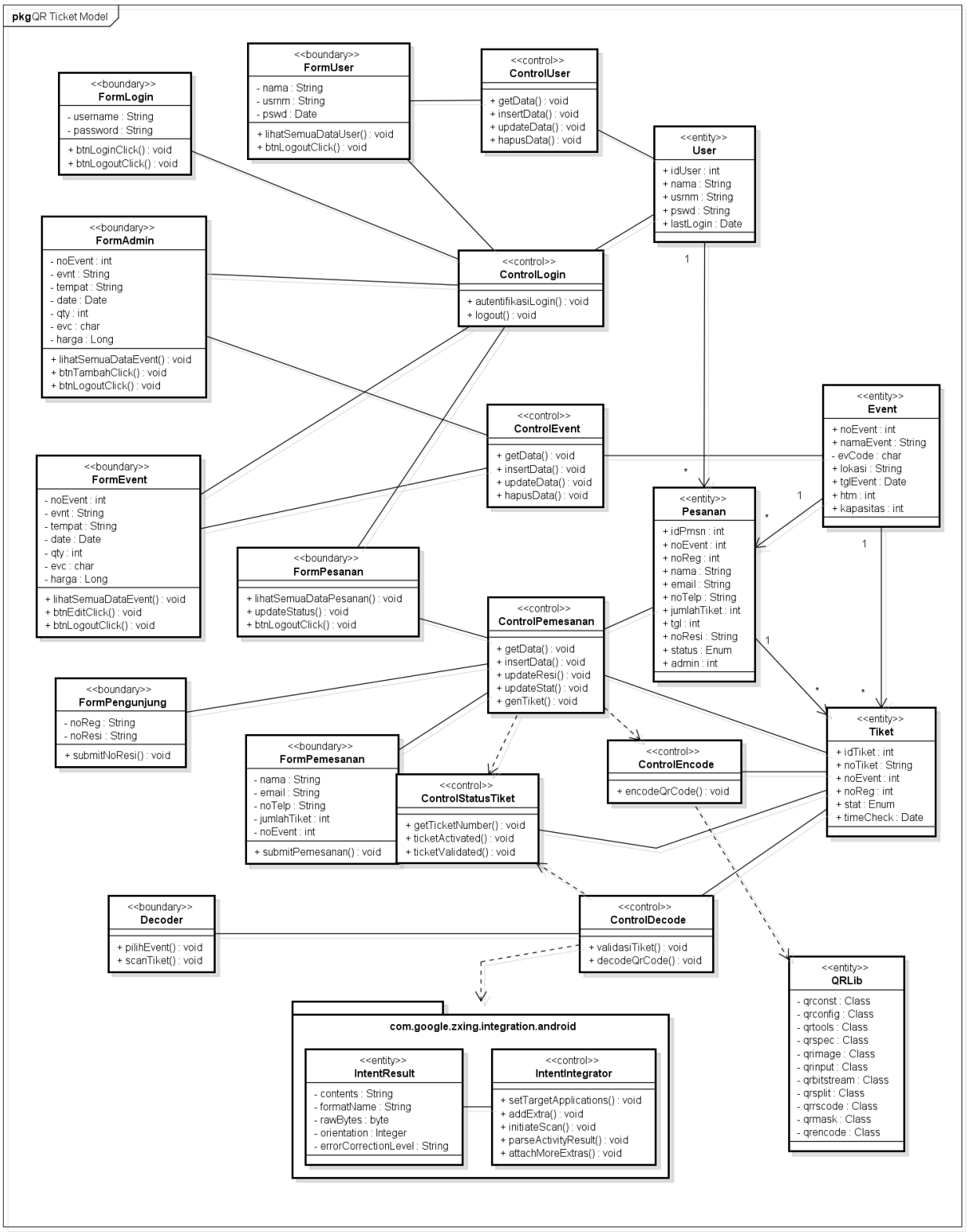
Pada proses *encoding* QR *code,* pengunjung memasukan nomor registrasi dan nomor resi pembayaran. Kemudian sistem akan melakukan validasi. Apabila nomor registrasi valid, maka sistem akan melakukan *encoding* tiket yang berbentuk QR *code* dengan status non-valid dan menampilkannya kepada pengunjung.



Gambar 3.23 Sequence Encode QR Code

3.6.2 Class Diagram

Terdapat tiga jenis kelas dalam sistem, yaitu kelas *Boundary*, kelas *Control*, dan kelas *Entity*. Kelas *Boundary* merupakan kelas yang digunakan untuk membangun tampilan antarmuka sistem. Kelas *Control* merupakan kelas yang digunakan sebagai tempat terjadinya seluruh proses yang ada dalam sistem dan juga validator yang memeriksa validasi data yang masuk, dan kelas *Entity* merupakan kelas yang digunakan sebagai penampung data atau representasi data itu sendiri. Seluruh kelas-kelas yang ada dalam sistem dapat dilihat pada Gambar 3.23.



Gambar 3.24 Class Diagram Sistem

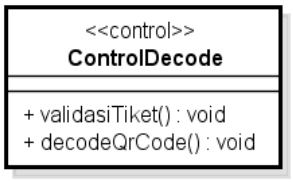
Setiap kelas pada *Class* Diagram Gambar 3.24 memiliki jenis kelasnya masing-masing. Untuk lebih lengkapnya seperti yang terlihat pada Tabel 3.18.

Tabel 3.18 Jenis-Jenis Kelas

No	Nama Kelas	Jenis Kelas
1	ControlDecode	Control
2	ControlEncode	Control
3	ControlEvent	Control
4	ControlLogin	Control
5	ControlPemesanan	Control
6	ControlStatusTiket	Control
7	ControlUser	Control
8	Decoder	Boundary
9	Event	Entity
10	FormAdmin	Boundary
11	FormLogin	Boundary
12	FormEvent	Boundary
13	FormPemesanan	Boundary
14	FormPesanan	Boundary
15	FromPengunjung	Boundary
16	FormUser	Boundary
17	Pesanan	Entity
18	QRLib	Entity
19	Tiket	Entity
20	User	Entity

3.6.2.1 Kelas ControlDecode

ControlDecode merupakan kelas yang melakukan verifikasi QR *code* dan melakukan *decoding* terhadap QR *code*. Gambar 3.25 merupakan kelas ControlDecode.



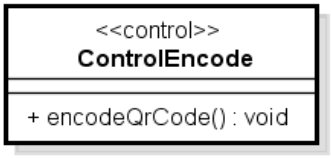
Gambar 3.25 Kelas ControlDecode

Kelas ini tidak memiliki atribut karena kelas ini hanya mengatur operasi yang dari kelas *Boundary* dengan kelas *Entity*. Berikut dekripsi dari operasi yang yang ada pada kelas ControlDecode.

1) *validasiTiket()*, operasi untuk melakukan validasi tiket.
2) *decodeQrCode()*, operasi untuk melakukan *decoding* QR *code* dari tiket.

3.6.2.2 Kelas ControlEncode

ControlEncode merupakan kelas yang melakukan *encoding* dari nomor tiket menjadi QR *code*. Gambar 3.26 merupakan gambar dari kelas ControlEncode.

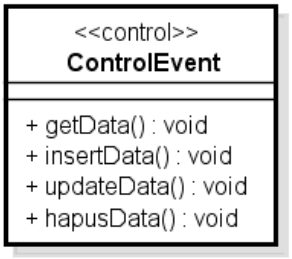


Gambar 3.26 Kelas ControlEncode

Sama seperti kelas *Control* lainnya. Pada kelas ControlEncode tidak terdapat atribut dan operasi yang ada pada kelas ini hanya satu, yaitu *encodeQrCode()* yang merupakan operasi untuk melakukan *encoding* QR *code*.

3.6.2.3 Kelas ControlEvent

ContolEvent merupakan kelas yang melakukan fungsional atau operasi pengelolaan event yang mana berhubungan dengan FormEvent yang dikelola oleh administrator. Gambar 3.27 merupakan gambar dari kelas ControlEvent.



Gambar 3.27 Kelas ControlEvent

Pada kelasi ControlEvent terdapat 4 operasi yang menghubungkan kelas *Boundary* FormEvent dengan kelas *Entity* Event. Berikut deskripsi dari operasi pada kelas ControlEvent.

1) *getData()*, mengambil seluruh data *event* dari *database*.
2) *insertData()*, mengisi data *event* kedalam *database.*
3) *updateData()*, mengnyunting data *event* yang ada pada *database.*
4) *hapusData()*, menghapus data *event* dari *database.*

3.6.2.4 Kelas ControlLogin

ControlLogin merupakan kelas yang melakukan autentifikasi login yang mana berfungsi untuk antentifikasi administrator. Gambar 3.28 merupakan gambar dari kelas ControlLogin.

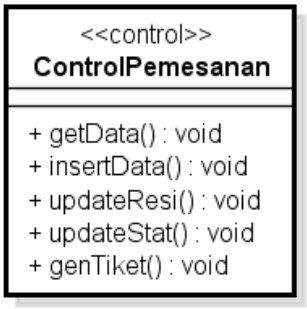


Gambar 3.28 Kelas ControlLogin

Pada kelas ControlLogin hanya terdapat satu operasi yaitu *autentifikasiLogin()*. Operasi ini merupalan operasi untuk autentifikasi login administrator kedalam sistem.

3.6.2.5 Kelas ControlPemesanan

ControlPemesanan merupakan kelas yang melakukan operasi dari 3 kelas *Boundary*, yaitu FormPesanan, FormPemesanan dan FormPengguna. Gambar 3.29 merupakan gambar dari kelas ControlPemesanan.



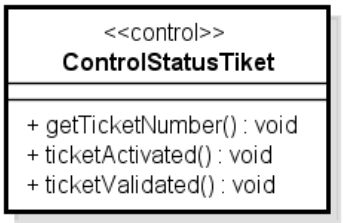
Gambar 3.29 Kelas ControlPemesanan

Pada kelasi ControlPemesanan terdapat 5 operasi yang menghubungkan kelas FormPesanan, FormPemesanan dan FormPengguna dengan kelas *Entity* Pesanan dan Tiket. Berikut deskripsi dari operasi yang ada pada kelas ControlPemesanan.

1) *getData(),* mengambil seluruh nilai data pemesanan dari *database* ke FormPesanan yang diakses oleh administrator.
2) *insertData(),* mengisi data pemesanan kedalam *database* dari pemesanan yang dilakukan oleh pengunjung.
3) *updateResi(),* melakukan update nomor resi pembayaran.
4) *updateStat(),* melakukan update status pembayaran yang dilakukan oleh administrator.
5) *genTiket(),* meng-*generate* tiket dan mengisinya kedalam *database* ketika pengunjung mensubmit nomor resi pembayaran.

3.6.2.6 Kelas ControlStatusTiket

ControlStatusTiket merupakan kelas yang digunakan oleh kelas ControlPemesanan dan kelas ControlDecode untuk melakukan *update* terhadap status tiket. Gambar 3.30 merupakan gambar dari kelas ControlStatusTiket.



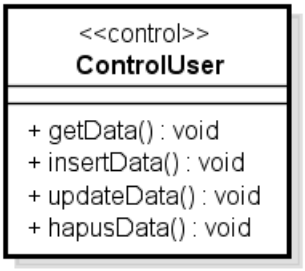
Gambar 3.30 Kelas ControlStatusTiket

Pada kelasi ControlStatusTiket terdapat 3 operasi. Berikut deskripsi dari operasi yang ada pada kelas ControlStatusTiket.

1) *getTicketNumber()*, mengambil nilai dari nomor tiket.
2) *ticketActivated()*, mengubah nilai dari status tiket menjadi valid setelah administrator melakukan *update* terhadap status pembayaran yang valid.
3) *ticketValidated()*, mengubah nilai dari status tiket dari valid menjadi *validated* setelah CotrolDecode melakukan proses *decoding* QR *code* dan verifikasi tiket.

3.6.2.7 Kelas ControlUser

ContolUser merupakan kelas yang melakukan fungsional atau operasi pengelolaan user yang mana berhubungan dengan FormUser yang dikelola oleh administrator. Gambar 3.31 merupakan gambar dari kelas ContolUser.



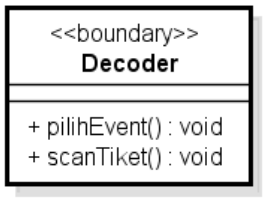
Gambar 3.31 Kelas ContolUser

Pada kelasi ContolUser terdapat 4 operasi yang menghubungkan kelas *Boundary* FormUser dengan kelas *Entity* User. Berikut deskripsi dari operasi pada kelas ContolUser.

1) *getData()*, mengambil seluruh data pengguna dari *database*.
2) *insertData()*, mengisi data pengguna kedalam *database.*
3) *updateData()*, mengnyunting data pengguna yang ada pada *database.*
4) *hapusData()*, menghapus data pengguna dari *database.*

3.6.2.8 Kelas Decoder

Decoder merupakan antarmuka dari aplikasi *mobile* yang berfungsi untuk melakukan *decoding* QR *code* dan verifikasi tiket yang dilakukan oleh pemeriksa tiket. Gambar 3.32 merupakan gambar dari kelas Decoder.

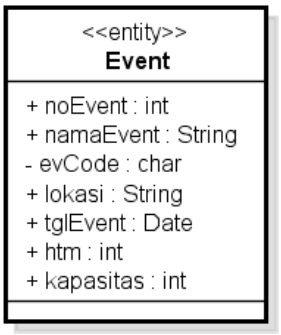


Gambar 3.32 Kelas Decoder

Pada kelas Decoder terdapat dua buah operasi. *pilihEvent()* berfungsi untuk memilih *event* yang sedang berlangsung sedangkan *scanTiket()* berfungsi untuk memanggil fungsi *decoding* QR *code* dari ControlDecode.

3.6.2.9 Kelas Event

Kelas Event merupakan kelas *Entity* yang berisi atribut yang berhubungan dengan data *event.* Gambar 3.33 merupakan gambar dari kelas Event.



Gambar 3.33 Kelas Event

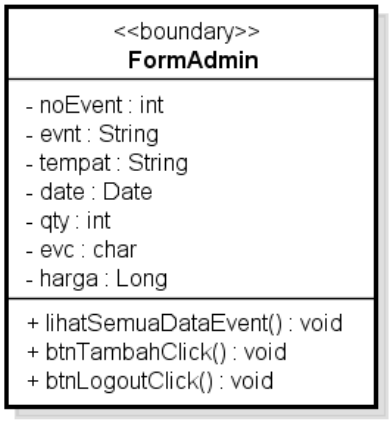
Pada kelas Event hanya terdapat atribut karena kelas Event merupakan kelas *Entity* dan semua operasi telah dilakukan oleh kelas ControlEvent. Pada Tabel 3.19 adalah deskripsi dari atribut yang ada pada kelas Event.

Tabel 3.19 Dekripsi Atribut pada Kelas Event

Nama	Tipe	Keterangan
noEvent	int	Nilai dari nomor *event*
namaEvent	String	Nilai dari nama *event*
evCode	char	Nilai dari kode *event*
lokasi	String	Nilai dari lokasi *event*
tglEvent	Date	Nilai dari tangggal diselenggarakannya *event*
htm	int	Nilai dari harga tiket *event*
kapasitas	int	Nilai dari kapasitas atau jumlah tiket maksimum

3.6.2.10 Kelas FormAdmin

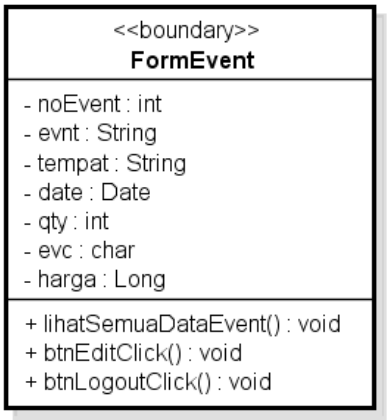
FormAdmin merupakan antarmuka dari halaman utama administrator. Pada halaman ini ditampilkan juga daftar *event* yang diselenggarakan.



Gambar 3.34 Kelas FormAdmin

3.6.2.11 Kelas FormEvent

FormEvent merupakan antarmuka dari pengelolaan data *event* yang dikelola oleh administrator. Dengan kata lain FormEvent merupakan tempat interaksi administrator dengan sistem dalam hal pengelolaan data *event.*

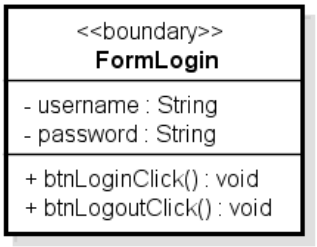


Gambar 3.35 Kelas FormEvent

Pada FromEvent terdapat satu operasi yang berfungsi untuk menampilkan seluruh data event yaitu *lihatSemuaDataEvent().* Operasi ini terhubung dengan operasi *getData()* pada kelas ControlEvent.

3.6.2.12 Kelas FormLogin

FormLogin merupakan antarmuka dari sistem untuk proses login administrator. FormLogin merupakan tempat interaksi administrator dengan sistem dalam hal autentifiksasi login.



Gambar 3.36 Kelas FormLogin

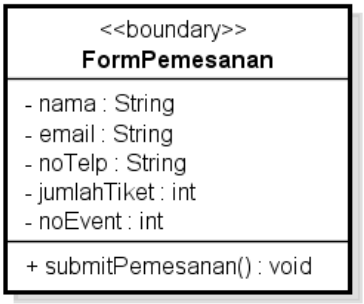
Pada FromLogin terdapat satu operasi *btnLoginClick()* untuk autentifikasi pengguna. Satu operasi lainnya yaitu *btnLogoutClick()*untuk fungsionalitas logout.

Tabel 3.20 Deskripsi Atribut pada FormLogin

Nama	Tipe	Keterangan
username	String	Menampung nilai username
password	String	Menampung nilai password

3.6.2.13 Kelas FormPemesanan

FormPemesanan merupakan antarmuka dari proses pemesanan yang dilakukan oleh pengunjung. FormPemesanan merupakan tempat interaksi pengunjung dengan sistem dalam proses pengisian data pemesanan tiket.



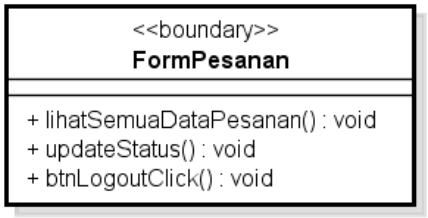
Gambar 3.37 Kelas FormPemesanan

Pada FromPemesanan terdapat satu operasi yang berfungsi untuk mengirimkan seluruh data pemesanan yaitu *submitPemesanan().* Operasi ini

mengirimkan data pemesanan pada kelas ControlPemesanan untuk kemudian dilakukan *insert* pada *database*.

3.6.2.14 Kelas FormPesanan

FormPesanan merupakan antarmuka dari pengelolaan data pemesanan yang dikelola oleh administrator. FormPesanan merupakan tempat interaksi administrator dengan sistem dalam hal melihat data pemesanan dan melakukan update status pembayaran.

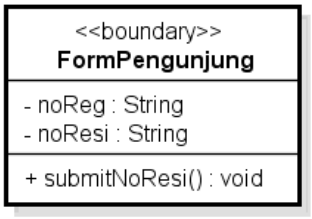


Gambar 3.38 Kelas FormPesanan

Pada FromPesanan terdapat satu operasi yang berfungsi untuk menampilkan seluruh data pesanan yaitu *lihatSemuaDataPesanan().* Operasi ini terhubung dengan operasi *getData()* pada kelas ControlPemesanan.

3.6.2.15 Kelas FormPengunjung

FormPengunjung merupakan halaman utama dari antarmuka sistem yang yang digunakan pengunjung. Dalam FormPengunjung terjadi interaksi pengunjung menginputkan nomor resi pembayaran sebagai tanda bahwa tiket telah dibayar.

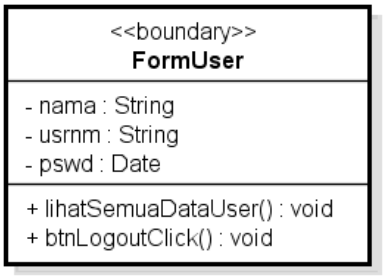


Gambar 3.39 Kelas FormPengunjung

Pada FromPengunjung terdapat satu operasi yang berfungsi untuk mengirimkan seluruh data pemesanan yaitu *submitNoResi().* Operasi ini mengirimkan data nomor resi pembayaran pada kelas ControlPemesanan untuk kemudian dilakukan *insert* pada *database*.

3.6.2.16 Kelas FormUser

FormUser merupakan antarmuka dari pengelolaan data pengguna yang dikelola oleh administrator. Dengan kata lain FormUser merupakan tempat interaksi administrator dengan sistem dalam hal pengelolaan data pengguna.

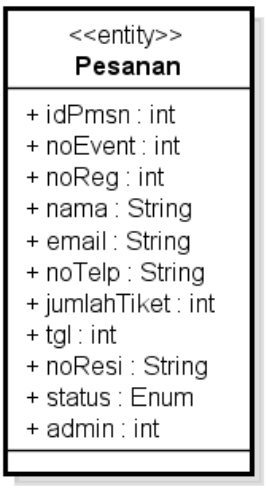


Gambar 3.40 Kelas FormUser

Pada FromUser terdapat satu operasi yang berfungsi untuk menampilkan seluruh data event yaitu *lihatSemuaDataUser().* Operasi ini terhubung dengan operasi *getData()* pada kelas ControlUser.

3.6.2.17 Kelas Pesanan

Kelas Pesanan merupakan kelas *Entity* yang berisi atribut yang berhubungan dengan data pemesanan. Gambar 3.41 merupakan gambar dari kelas Pesanan.



Gambar 3.41 Kelas Pesanan

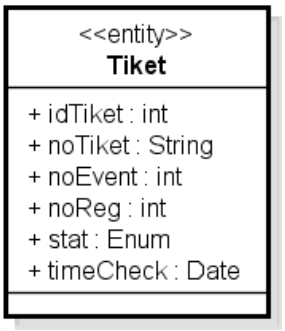
Pada kelas Pesanan hanya terdapat atribut karena kelas Pesanan merupakan kelas *Entity* dan semua operasi telah dilakukan oleh kelas ControlPemesanan. Pada Tabel 3.21 adalah deskripsi dari atribut yang ada pada kelas Pesanan.

Tabel 3.21 Deskripsi Atribut pada Kelas Pesanan

Nama	Tipe	Keterangan
idPmsn	int	Nilai dari id pemesanan
noEvent	int	Nilai dari nomor *event*
noReg	String	Nilai dari nomor registrasi
nama	String	Nilai dari nama pengunjung yang memesan
email	String	Nilai dari email pengunjung yang memesan
noTelp	String	Nilai dari nomor telepon pengunjung
jumlahTiket	int	Nilai dari jumlah tiket yang dipesan
tgl	Date	Nilai dari tangggal pemesanan tiket
noResi	int	Nilai dari nomor resi pembayaran
status	Enum	Nilai dari status pembayaran
admin	int	Nilai dari admin yang memeriksa pembayaran

3.6.2.18 Kelas Tiket

Kelas Tiket merupakan kelas *Entitiy* yang berisi atribut yang berhubungan dengan data tiket. Gambar 3.42 adalah gambar dari kelas Tiket.



Gambar 3.42 Kelas Tiket

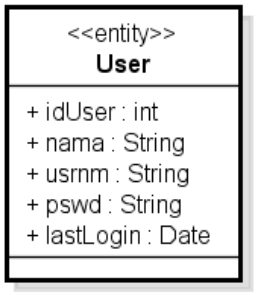
Pada kelas Tiket hanya terdapat atribut karena kelas Tiket merupakan kelas *Entity* dan semua operasi telah dilakukan oleh kelas *Control* yang berelasi. Pada Tabel 3.22 adalah deskripsi dari atribut yang ada pada kelas Tiket.

Tabel 3.22 Deskripsi Atribut pada Kelas Tiket

Nama	Tipe	Keterangagan
noTiket	String	Nilai dari nomor tiket
noEvent	int	Nilai dari nomor event
noReg	int	Nilai dari nomor registrasi pengunjung
stat	Enum	Nilai dari status tiket
timeCheck	Date	Nilai dari waktu validasi tiket

3.6.2.19 Kelas User

Kelas User merupakan kelas *Entitiy* yang berisi atribut yang berhubungan dengan data pengguna. Gambar 3.43 adalah gambar dari kelas User.



Gambar 3.43 Kelas User

Pada kelas User hanya terdapat atribut karena kelas User merupakan kelas *Entity* dan semua operasi telah dilakukan oleh kelas *Control* yang berelasi. Pada Tabel 3.23 adalah deskripsi dari atribut yang ada pada kelas User.

Tabel 3.23 Deskripsi Atribut pada Kelas User

Nama	Tipe	Keterangan
idUser	int	Nilai dari id *user*
nama	String	Nilai dari nama *user* administrator
usrnm	String	Nilai dari username *user* administrator
pswd	String	Nilai dari password *user* administrator
lastLogin	Date	Nilai dari waktu login terakhir administrator

3.6.3 Activity Diagram

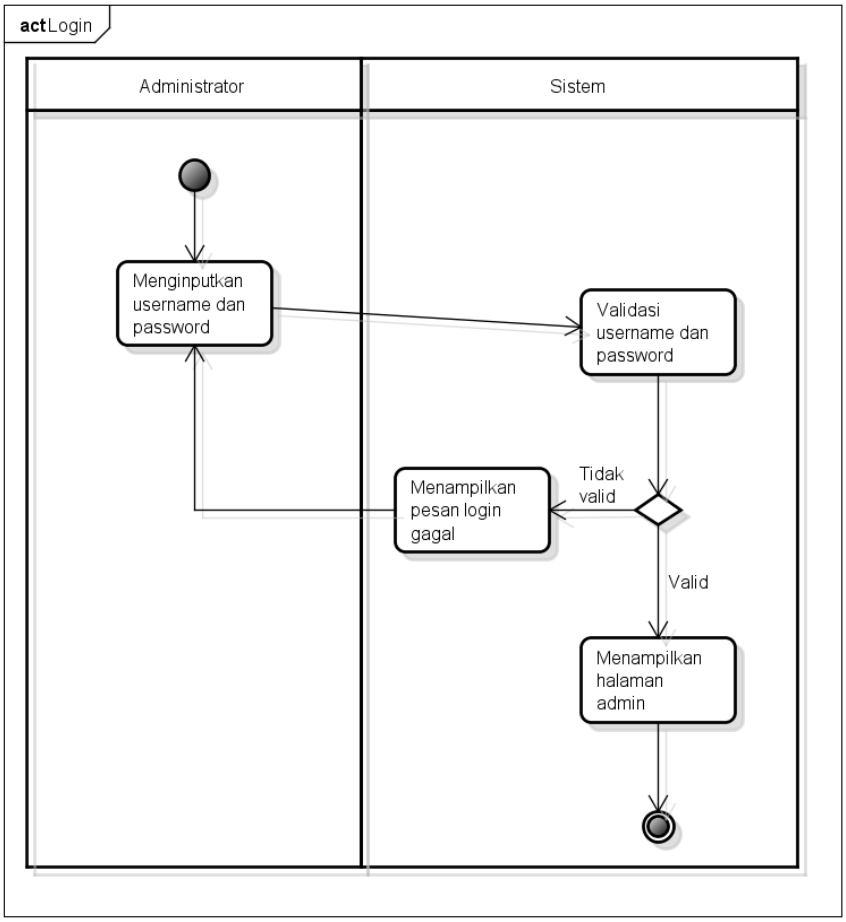
Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity* diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

3.6.3.1 Aktifitas Login

Aktifitas yang terdapat dalam login adalah melakukan autentifikasi pada login administrator. Dengan melakukan autentifikasi, sistem dapat memberikan akses kepada administrator untuk mengelola sistem. Tahapan tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Menginputkan username dan password, pada aktifitas ini, administrator menginputkan username dan password ke dalam sistem.
2) Validasi username dan password, pada aktifitas ini sistem akan melakukan validasi dan autentifikasi pengguna untuk memberikan akes kepada administrator.
3) Jika valid, maka sistem akan menampilkan halaman admin. Sedangkan jika tidak valid sistem akan menampilkan pesan login gagal.
4) Menampilkan pesan login, dalam aktifitas ini sistem akan menampilkan pesan login gagal.
5) Menampilkan halaman admin, aktifitas ini menandakan bahwa autentifikasi berhasil sehingga administrator dapat mengakses sistem.

Seluruh aktifitas login dapat dilihat tahapannya pada diagram aktifitas login pada Gambar 3.44.



Gambar 3.44 Diagram Aktifitas Login

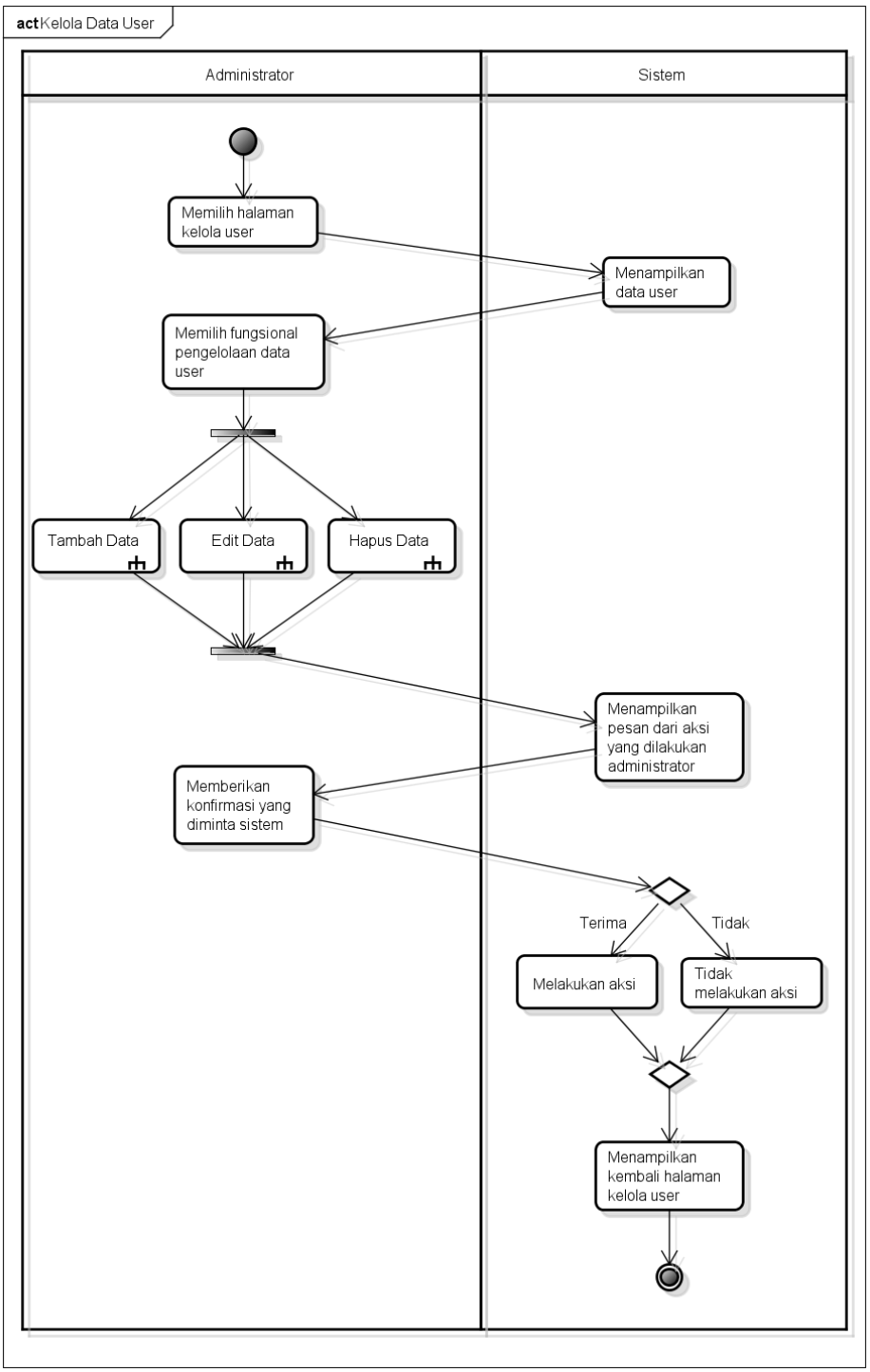
3.6.3.2 Aktifitas Kelola Data Pengguna

Aktifitas yang terdapat dalam kelola data pengguna adalah melakukan pengelolaan terhadap data pengguna. Dalam aktifitas ini, terdapat tiga sub aktifitas untuk proses pengelolaan data, diantaranya tambah data, edit data dan hapus data. Tahapan –tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memlilih halaman kelola data pengguna, aktifitas diawali dengan memilih halaman kelola data pengguna.
2) Menampilkan data pengguna, aktifitas ini sistem menampilkan seluruh data pengguna ketika sistem menampilkan halaman kelola data pengguna.
3) Memilih fungsional pengelolaan data pengguna, administrator memlilih fungsional dari tambah data, edit data dan hapus data.

4) Jika memilih tambah data, maka sistem akan menampilkan fungsional penambahan data.
5) Jika memilih edit data, maka sistem akan menampilkan fungsional edit data.
6) Jika memilih hapus data, maka sistem akan menghapus data yang dipilih
7) Menampilkan pesan sesuai aksi yang dipilih administrator, pada aktifitas ini sistem akan menampilkan pesan berupa konfirmasi untuk melakukan aksi sesuai dengan fungsional yang dipilih oleh administrator.
8) Memberikan konfirmasi yang diminta sistem, administrator membarikan konfirmasi untuk melakukan aksi.
9) Jika ya, maka sistem akan melanjutkan aksi yang diminta oleh administrator.
10) Jika tidak, maka sistem tidak akan melakukan aksi yang diminta oleh administrator.
11) Menampilkan kembali halaman kelola pengguna, pada aktifitas ini sistem akan menampilkan kembali halaman pengelolaan data pengguna.

Seluruh aktifitas kelola data *event* dapat dilihat tahapannya pada diagram aktifitas kelola data pengguna pada Gambar 3.45.



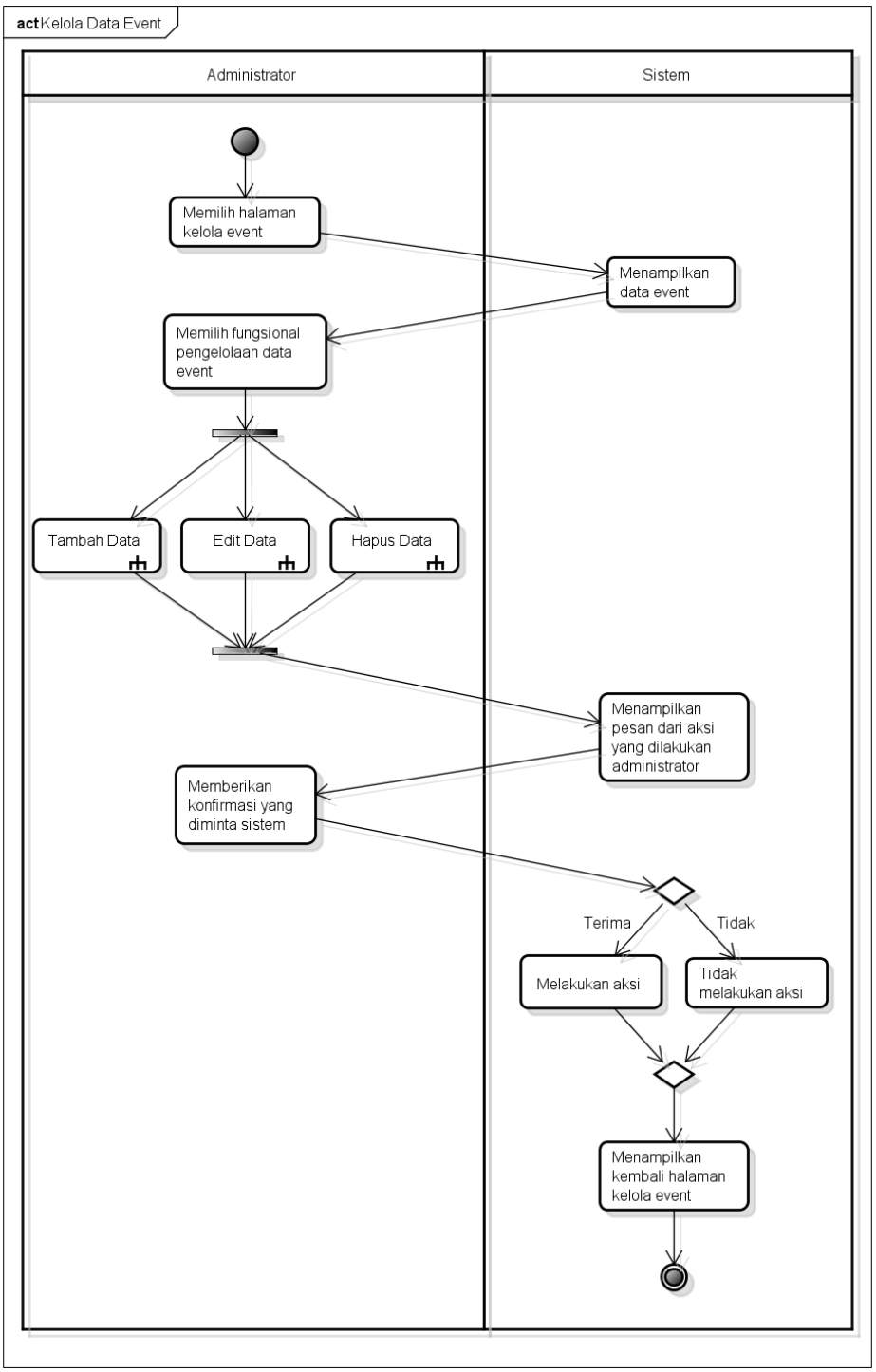
Gambar 3.45 Diagram Aktifitas Kelola Data Pengguna

3.6.3.3 Aktifitas Kelola Data Event

Aktifitas yang terdapat dalam kelola data *event* adalah melakukan pengelolaan terhadap data *event*. Dalam aktifitas ini, terdapat tiga sub aktifitas untuk proses pengelolaan data, diantaranya tambah data, edit data dan hapus data. Tahapan –tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memlilih halaman kelola data *event,* aktifitas diawali dengan memilih halaman kelola data *event*.
2) Menampilkan data *event,* aktifitas ini sistem menampilkan seluruh data *event* ketika sistem menampilkan halaman kelola data *event*.
3) Memilih fungsional pengelolaan data *event,* administrator memlilih fungsional dari tambah data, edit data dan hapus data.
4) Jika memilih tambah data, maka sistem akan menampilkan fungsional penambahan data.
5) Jika memilih edit data, maka sistem akan menampilkan fungsional edit data.
6) Jika memilih hapus data, maka sistem akan menghapus data yang dipilih
7) Menampilkan pesan sesuai aksi yang dipilih administrator, pada aktifitas ini sistem akan menampilkan pesan berupa konfirmasi untuk melakukan aksi sesuai dengan fungsional yang dipilih oleh administrator.
8) Memberikan konfirmasi yang diminta sistem, administrator membarikan konfirmasi untuk melakukan aksi.
9) Jika ya, maka sistem akan melanjutkan aksi yang diminta oleh administrator.
10) Jika tidak, maka sistem tidak akan melakukan aksi yang diminta oleh administrator.
11) Menampilkan kembali halaman kelola *event,* pada aktifitas ini sistem akan menampilkan kembali halaman pengelolaan data *event*.

Seluruh aktifitas kelola data *event* dapat dilihat tahapannya pada diagram aktifitas kelola data *event* pada Gambar 3.46.



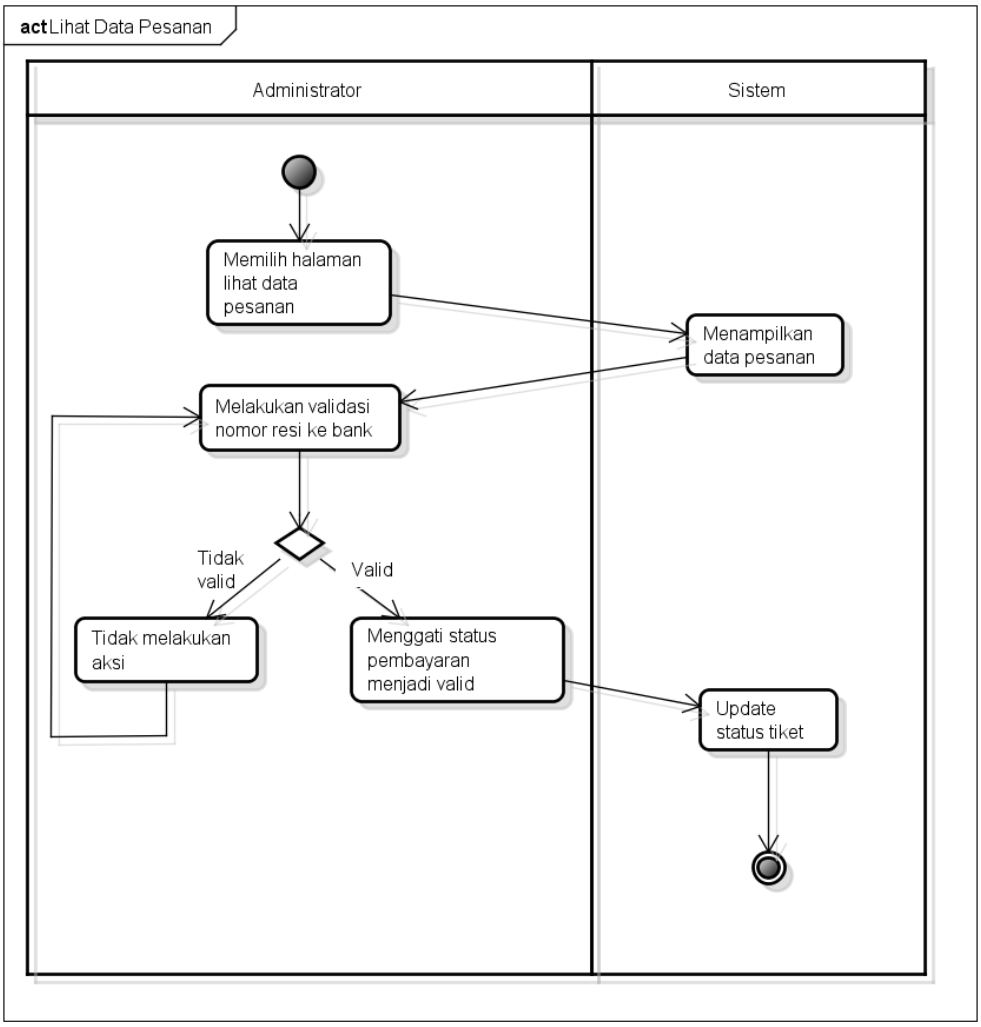
Gambar 3.46 Diagram Aktifitas Kelola Data Event

3.6.3.4 Aktifitas Lihat Data Pesanan

Aktifitas yang terdapat pada lihat data pemesanan adalah administrator memilih halaman lihat data pesanan dan sistem akan menampilkan halaman tersebut. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memlilih halaman lihat data pesanan, aktifitas dimulai dari administrator yang memilih halaman lihat data pesanan.
2) Menampilkan data pesanan, pada aktifitas ini sistem menampilkan halaman liaht data pesanan bersama dengan menampilkan data pesanan.
3) Melakukan validasi nomor resi ke bank, aktifitas ini administrator akan melakukan validasi dari resi pembayaran ke bank yang bersangkutan secara manual. Bisa melalui *e-banking* atau dengan panggilan telepon.
4) Jika nomor resi pembayaran valid, maka administrator akan mengupdate status pembayaran menjadi valid.
5) Jika nomor resi pembayaran tidak valid, maka administrator tidak akan melakukan aksi.
6) Update status tiket, pada aktifitas ini, sistem akan melakukan aktifitas update status tiket apabila administrator mengupdate status pembayaran.

Seluruh aktifitas lihat data pesanan dapat dilihat tahapannya pada diagram aktifitas lihat data pesanan pada Gambar 3.47.



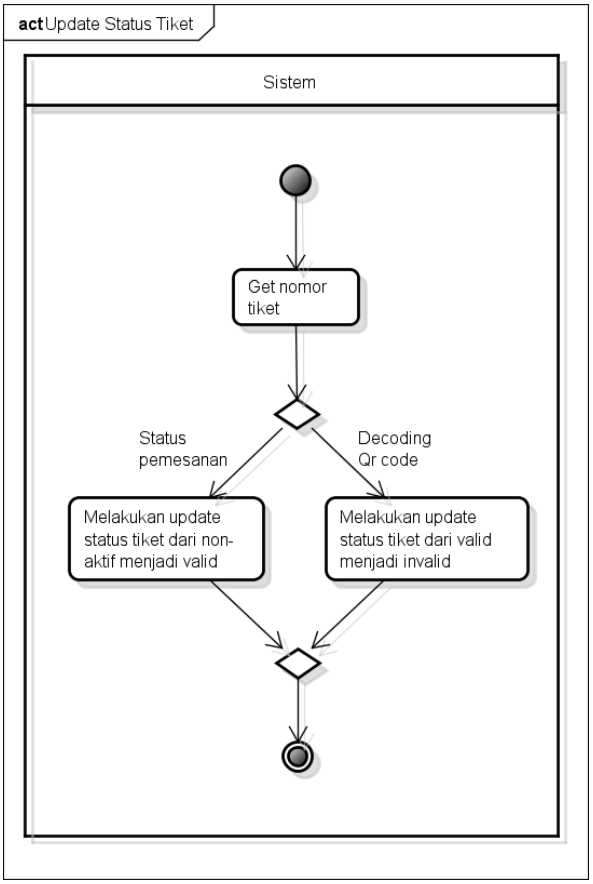
Gambar 3.47 Diagram Aktifitas Lihat Data Pesanan

3.6.3.5 Aktifitas Update Status Tiket

Aktifitas yang terdapat dalam sistem adalah melakukan *update* pada status tiket dari nonaktif menjadi valid. Hal ini untuk membuat tiket dapat verifikasi oleh *decoder* sehingga pengunjung mendapatkan akses ke area *event*. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Get nomot tiket, aktifitas ini sistem mengambil nilai dari nomor tiket pada proses *update* status pembayaran.
2) Jika yang terjadi sebelumnya adalah proses update pembayaran, maka sistem akan mengganti status tiket menjadi valid.
3) Jika yang terjadi sebelumnya adalah proses *decoding* QR *code*, maka sistem akan mengganti staus tiket menjadi *invalid* atau *activated.*

Seluruh aktifitas *update* status tiket dapat dilihat tahapannya pada diagram aktifitas *update* status tiket pada Gambar 3.48.



Gambar 3.48 Diagram Aktifitas Update Status Tiket

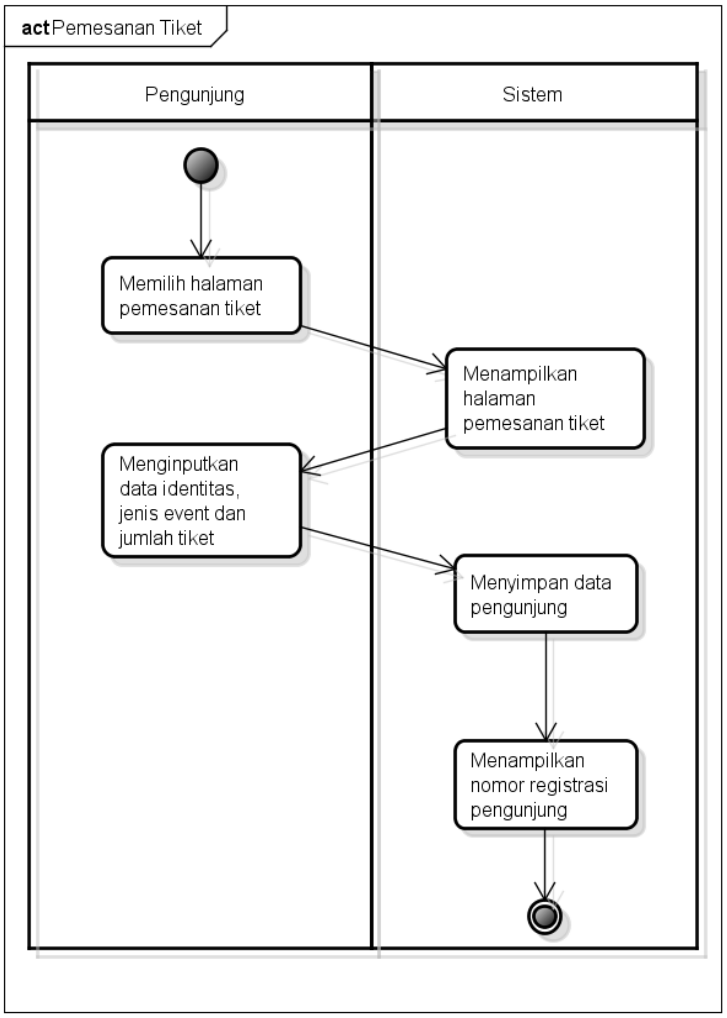
3.6.3.6 Aktifitas Pemesanan Tiket

Aktifitas yang terdapat dalam pemesanan tiket adalah pengunjung memilih halaman pemesanan tiket dan melakukan pengisian data pada form pemesanan tiket. Kemudian sistem menyimpan data pemesanan. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memilih halaman pemesanan tiket, pengunjung memilih halaman pemesanan tiket.
2) Menampilkan halaman pemesanan tiket, sistem menampilkan halaman pemesanan tiket.
3) Menginputkan data identitas, jenis *event* dan jumlah tiket, pengunjung memasukan data tersebut dan menekan tombol submit.

4) Menyimpan data pengunjung, sistem menyimpan data pengunjung dan membangkitkan tiket, kemudiannya menyimpan tiket tersebut.
5) Menampilkan nomor registrasi, sistem menampilkan nomor registrasi pengunjung untuk digunakan ketika mengirim nomor resi pembayaran.

Seluruh aktifitas pemesanan tiket dapat dilihat tahapannya pada diagram aktifitas pemesanan tiket pada Gambar 3.49.



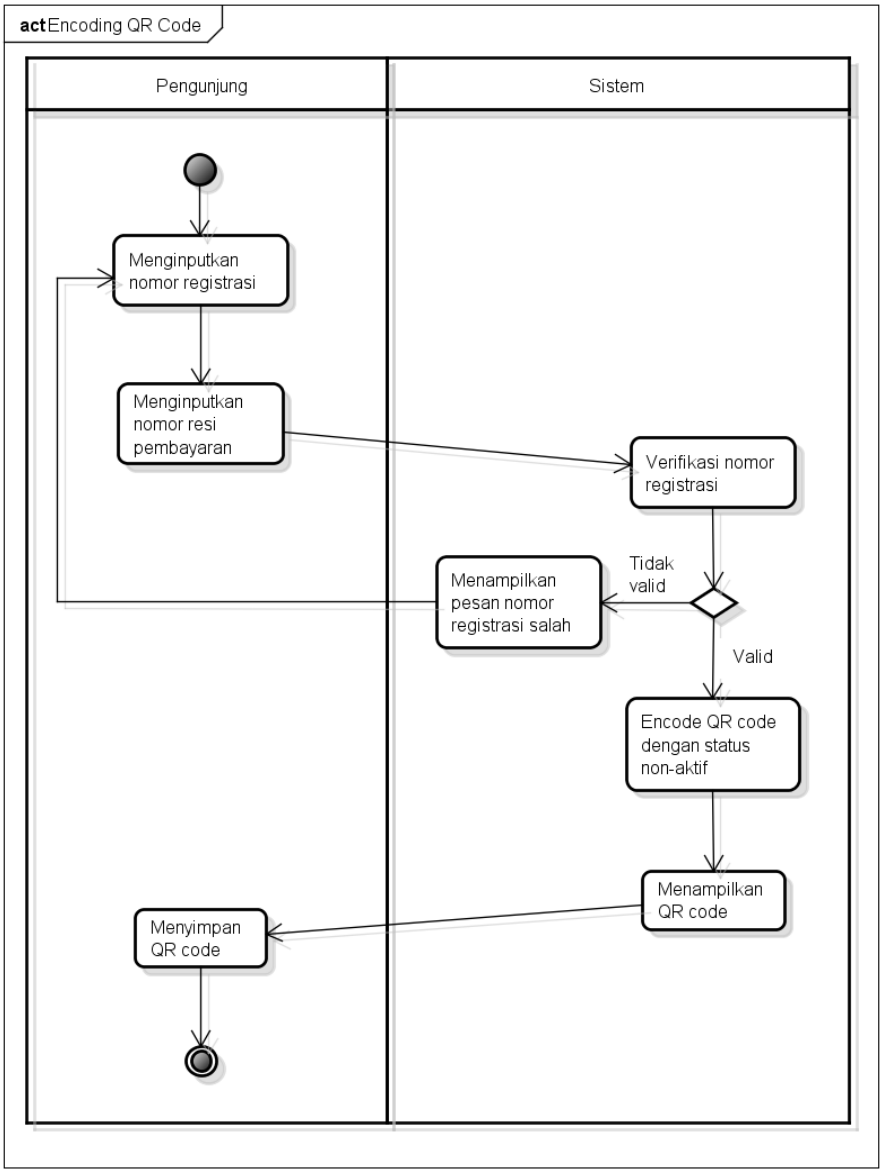
Gambar 3.49 Diagram Aktifitas Pemesanan Tiket

3.6.3.7 Aktifitas Encoding QR Code

Aktifitas yang terdapat dalam *encoding QR code* adalah melakukan *encoding* QR *code* setelah pengunjung menginputkan nomor resi pembayaran ke sistem. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Menginputkan nomor registrasi, pengunjung menginputkan nomor registrasi.
2) Menginputkan nomor resi, pengunjung menginputkan nomor resi pembayaran.
3) Verifikasi nomor registrasi, sistem akan melakukan verifikasi nomor registrasi dari pengunjung.
4) Jika nomor registrasi valid, maka sistem akan melakukan *encoding* QR *code* yang mana di *generate* dari nomor tiket yang telah disimpan sistem ketika pengunjung melakukan pemesanan tiket.
5) Jika nomor registrasi tidak valid, maka sistem akan menampilkan bahwa nomor registrasi salah dan pengunjung akan menginputkan kembali nomor registrasi dan nomor resi pembayaran.
6) Menampilkan QR *code*, sistem akan menampilkan QR *code* hasil dari proses *encoding* yang merupakan tiket.
7) Menyimpan QR *code*, pengunjung menyimpan QR *code* tersebut, bisa disimpan dalam bentuk file, di foto atau langsung di cetak pada kertas. QR *code* ini nantinya akan menjadi tiket masuk bagi pengunjung.

Seluruh aktifitas *encoding* QR *code* dapat dilihat tahapannya pada diagram aktifitas *encoding* QR *code* pada Gambar 3.50.



Gambar 3.50 Diagram Aktifitas Encoding QR Code

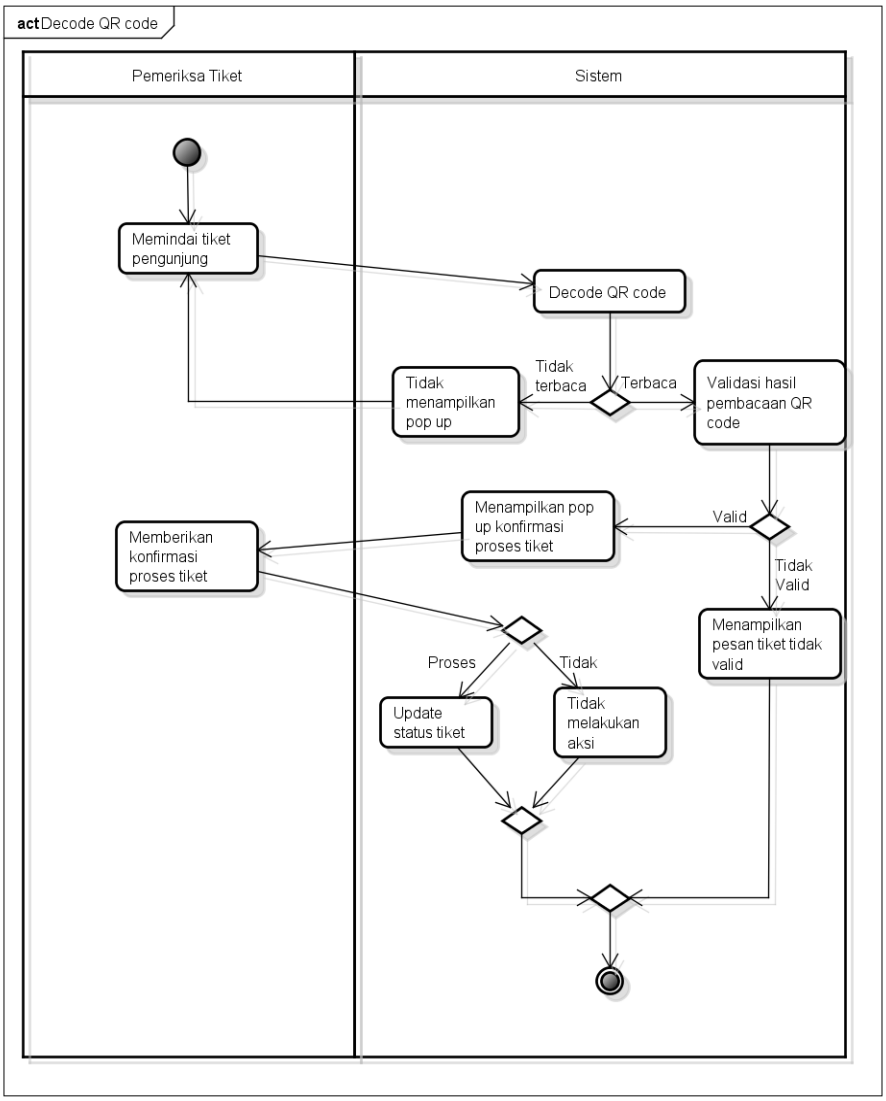
3.6.3.8 Aktifitas Decoding QR Code

Aktifitas yang ter dapat dalam *decoding* QR *code* adalah melakukan *decoding* dan verifikasi dari tiket yang berbentuk QR *code.* Aktifitas ini dilakukan oleh pemeriksa tiket. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memindai tiket pengunjung, pemeriksa tiket memindai tiket pengunjung dengan cara menangkap gambar menggunakan kamera yang ada pada perangkat *mobile* yang merupakan *decoder*.

2) *Decode* QR *code*, sistem melakukan *decoding* QR *code* yaitu proses penguraian isi dari citra pada QR *code*.
3) Jika tiket terbaca, sistem akan melakukan validasi hasil pembacaan QR *code* ke server untuk mengetahui apakah tiket ini valid.
4) Jika tiket valid, sistem akan meminta konfirmasi untuk memproses tiket.
5) Memberikan konfirmasi proses tiket, pemeriksa tiket akan memberikan konfirmasi untuk proses tiket.
6) Jika ya, maka sistem akan melakukan aktifitas *update* status tiket.
7) Jika tidak, maka sitem tidak akan melakukan aksi.
8) Jika tiket tidak valid, sistem akan menampilkan pesan bahwa tiket tidak valid.
9) Jika tiket tidak terbaca, sistem tidak akan bereaksi dan pemeriksa tiket akan melakukan pemindaian tiket lainnya.

Seluruh aktifitas *decoding* QR *code* dapat dilihat tahapannya pada diagram aktifitas *decoding* QR *code* pada Gambar 3.51.



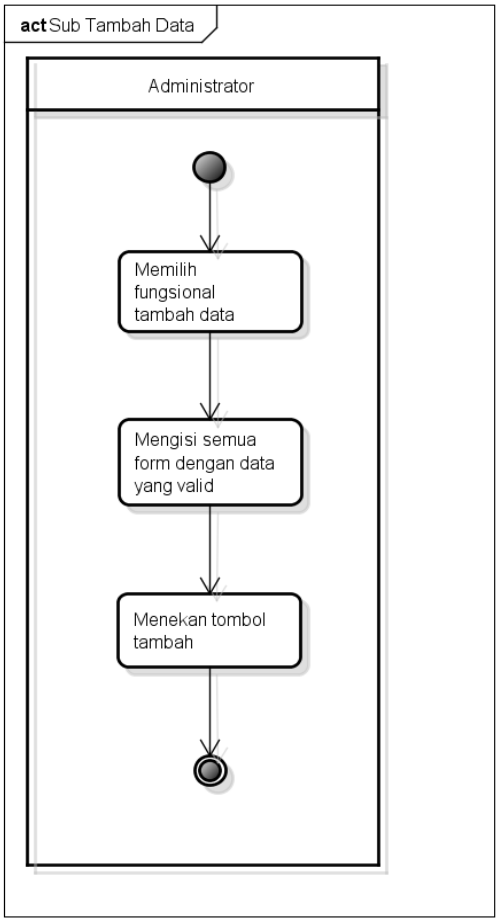
Gambar 3.51 Diagram Aktifitas Decoding QR Code

3.6.3.9 Sub Aktifitas Tambah Data

Aktifitas yang terdapat dalam sub aktifitas tambah data adalah melakukan *insert* data. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memilih fungsional tambah data, administrator memilih fungsional tambah data.
2) Mengisi semua form dengan data yang valid, administrator mengisi semua form yang ada pada sistem dengan data yang valid.
3) Menekan tombol tambah, administrator menekan tombol tambah.

Seluruh aktifitas tambah data dapat dilihat tahapannya pada diagram sub aktifitas tambah data pada Gambar 3.52.



Gambar 3.52 Diagram Sub Aktifitas Tambah Data

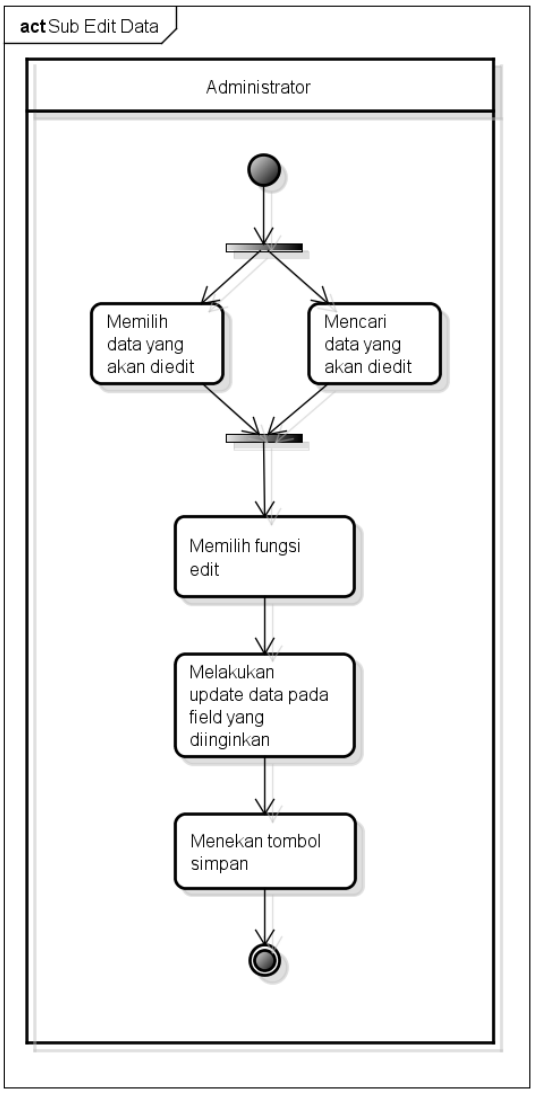
3.6.3.10 Sub Aktifitas Edit Data

Aktifitas yang terdapat dalam sub aktifitas edit data adalah melakukan *update* pada data. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memilih data yang akan diedit, administrator memilih data yang akan diedit.
2) Mencari data yang akan diedit, administrator mencari data yang akan diedit.
3) Memilih fungsional edit data, administrator memilih fungsional edit data.
4) Melakukan *update* data pada field yang diinginkan, administrator mengupdate field yang diinginkan dengan data yang valid.

5) Menekan tombol simpan, administrator menekan tombol simpan.

Seluruh aktifitas edit data dapat dilihat tahapannya pada diagram sub aktifitas edit data pada Gambar 3.53.



Gambar 3.53 Diagram Sub Aktifitas Edit Data

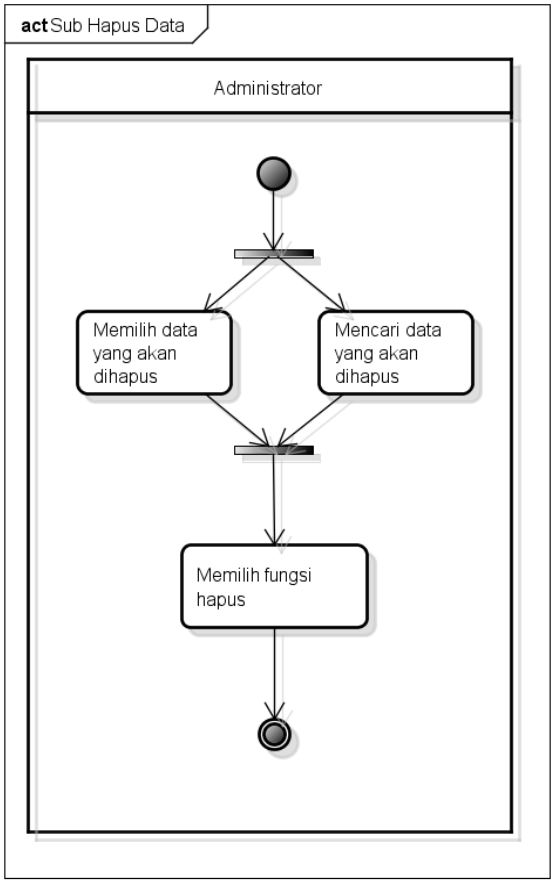
3.6.3.11 Sub Aktifitas Hapus Data

Aktifitas yang terdapat dalam sub aktifitas edit data adalah melakukan *update* pada data. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut.

1) Memilih data yang akan dihapus, administrator memilih data yang akan dihapus.

2) Mencari data yang akan dihapus, administrator mencari data yang akan dihapus.
3) Memilih fungsional hapus data, administrator memilih fungsi hapus data.

Seluruh aktifitas edit data dapat dilihat tahapannya pada diagram sub aktifitas edit data pada Gambar 3.54.



Gambar 3.54 Diagram Sub Aktifitas Hapus Data

3.7 Perancangan Basis Data

Perancangan basis data berguna untuk memenuhi kebutuhan-kebutuhan konten informasi dan menyediakan struktur informasi yang alami dan mudah dipahami. Perancangan basis data terdiri dari skema relasi, diagram relasi dan struktur file.

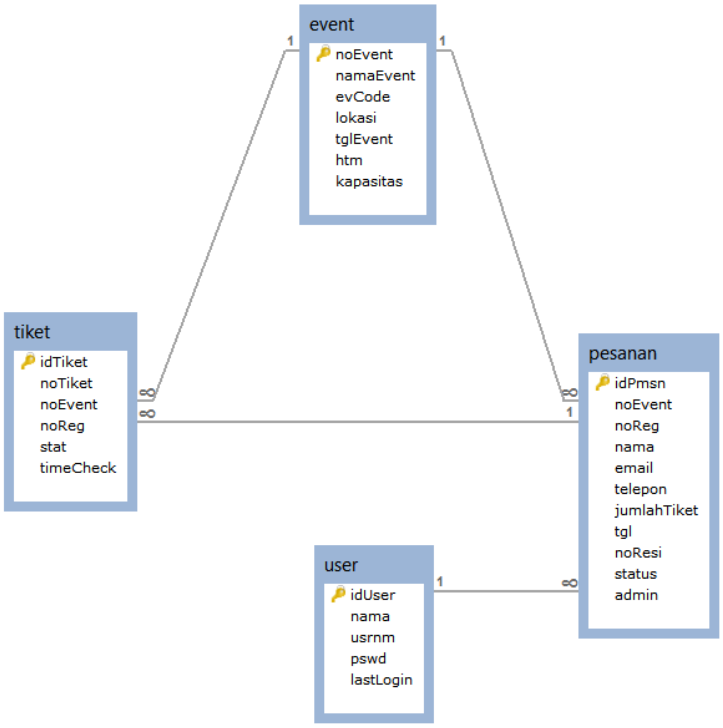
3.7.1 Skema Relasi

Skema relasi merupakan notasi untuk menggambar tabel, atribut tiap tabel dan hubungan atribut kunci antar tabel yang berelasi. Berikut adalah skema relasi dari basis data pada sistem yang dibangun.

1) User = (idUser, nama, usrnm, pswd, lastLogin)
2) Event = (noEvent, namaEvent, evCode, lokasi, tglEvent, htm, kapasitas)
3) Pesanan = (idPmsn, noEvent, noReg, nama, email, noTelp, jumlahTiket, tgl, noResi, status, admin)
4) Tiket = (idTiket, noTiket, noEvent, noReg, stat, timeChecked)

3.7.2 Diagram Relasi

Diagram relasi merupakan diagram yang menjelaskan realasi antar tabel atau entitas. Gambar 3.55 adalah diagram relasi dari basis data pada sistem yang dibangun.



Gambar 3.55 Diagram Relasi

Dari Gambar 3.55 dapat terlihat relasi dari setiap entitas. Angka yang ada menunjukan kardinalitas antar entitas.

3.7.3 Struktur File

Struktur file menggambarkan file yang akan mewakili satu tabel. Dimana jumlah tabel maupun atribut pada struktur file harus sama dengan yang ada di skema relasi. Berikut adalah struktur file dari basis data pada sistem yang dikembangkan.

1) Tabel User

Tabel 3.24 Struktur File Tabel User

Nama Field	Tipe	Panjang	Kunci	Keterangan
idUser	int	6	Primary key	Auto Increment
nama	varchar	50	-	-
usrnm	varchar	20	-	-
pswd	char	32	-	-
lastLogin	datetime	-	-	-

2) Tabel Event

Tabel 3.25 Struktur File Tabel Event

Nama Field	Tipe	Panjang	Kunci	Keterangan
noEvent	int	6	Primary key	Auto increment
namaEvent	varchar	100	-	-
evCode	char	3	-	Unique
lokasi	varchar	255	-	-
tglEvent	datetime	-	-	-
htm	int	7	-	-
kapasitas	int	6	-	-

3) Tabel Pesanan

Tabel 3.26 Struktur File Tabel Pesanan

Nama Field	Tipe	Panjang	Kunci	Keterangan
idPmsn	int	6	Primary key	Auto increment
noEvent	int	6	Foreign key	FK dari noEvent pada tabel Event
noReg	varchar	25	-	Unique
nama	varchar	50	-	-
email	varchar	100	-	-
telp	varchar	15	-	-
jumlahTiket	int	3	-	-
tgl	datetime	-	-	-
noResi	varchar	25	-	-
status	enum	11	-	"Belum Lunas", "Lunas"
admin	int	6	-	FK dari idUser pada tabel User

4) Tabel Tiket

Tabel 3.27 Struktur File Tabel Tiket

Nama Field	Tipe	Panjang	Kunci	Keterangan
idTiket	int	6	Primary key	Auto increment
noTiket	varchar	8	-	Unique
noEvent	int	6	Foreign key	FK dari noEvent pada tabel Event
noReg	varchar	25	Foreign key	FK dari noReg pada tabel pesanan

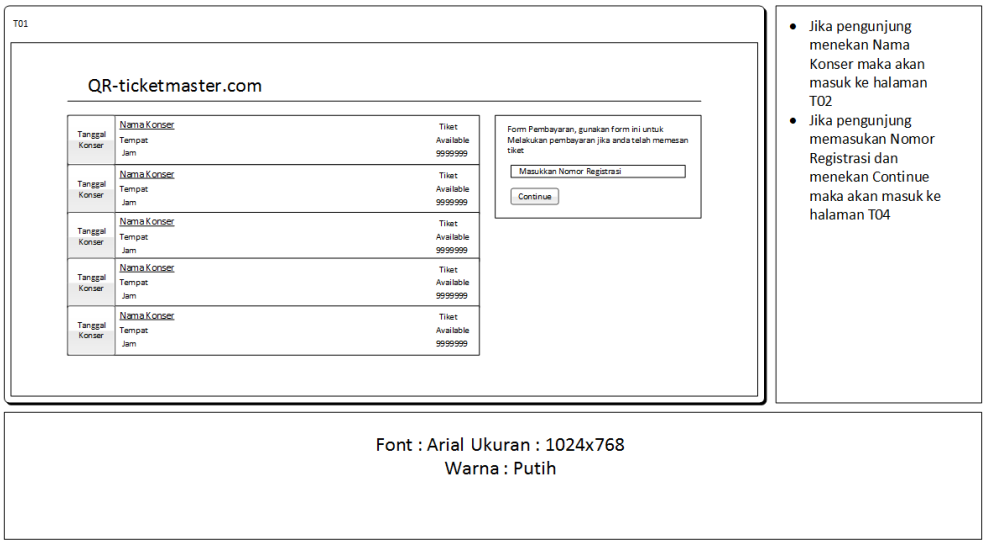
stat	enum	13	-	"Non-aktif", "Aktif", "Terverifikasi"
timeCheck	datetime	-	-	-

3.8 Perancangan Antarmuka

Perancangan antarmuka merupakan perancangan dari antarmuka sistem dimana antamuka merupakan tempat sistem berinteraksi dengan pengguna. Dalam perancangan antarmuka sistem ini, terdapat dua jenis antarmuka, yaitu antarmuka form atau halaman dan antarmuka pesan.

3.8.1 Antarmuka Halaman Pengunjung (T01)

Halaman pengunjung merupakan halaman utama yang diakses pengunjung. Adapun perancangan antarmuka untuk halaman pengunjung yang dibangun adalah seperti pada Gambar 3.56 berikut.



Gambar 3.56 Antarmuka Halaman Pengunjung

3.8.2 Antarmuka Halaman Pemesanan Tiket (T02)

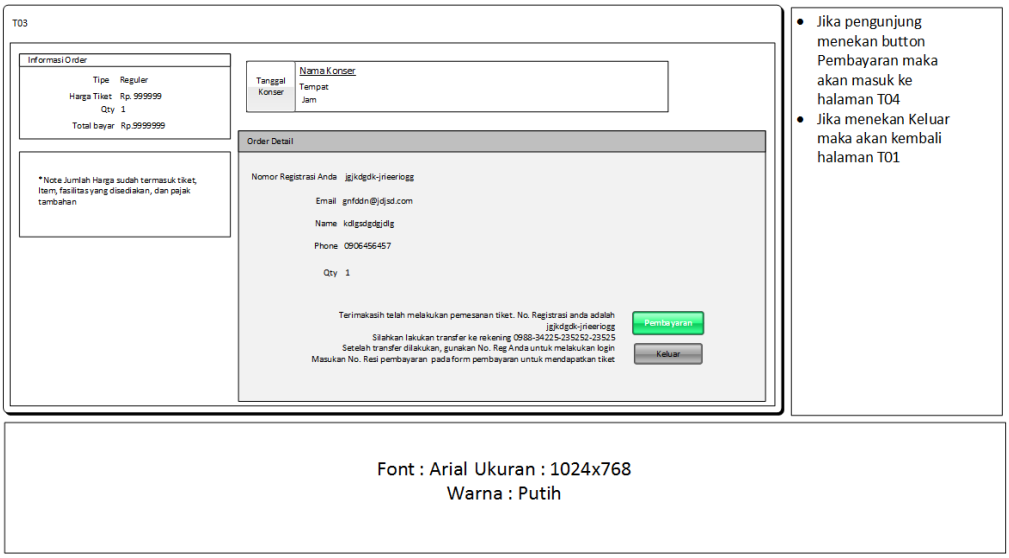
Halaman pemesanan tiket merupakan halaman dimana pengunjung dapat melakukan pemesanan tiket. Adapun perancangan antarmuka untuk halaman pemesanan tiket yang dibangun adalah seperti Gampar 3.57 berikut.



Gambar 3.57 Antarmuka Halaman Pemesanan Tiket

3.8.3 Antarmuka Halaman Detail Pemesanan (T03)

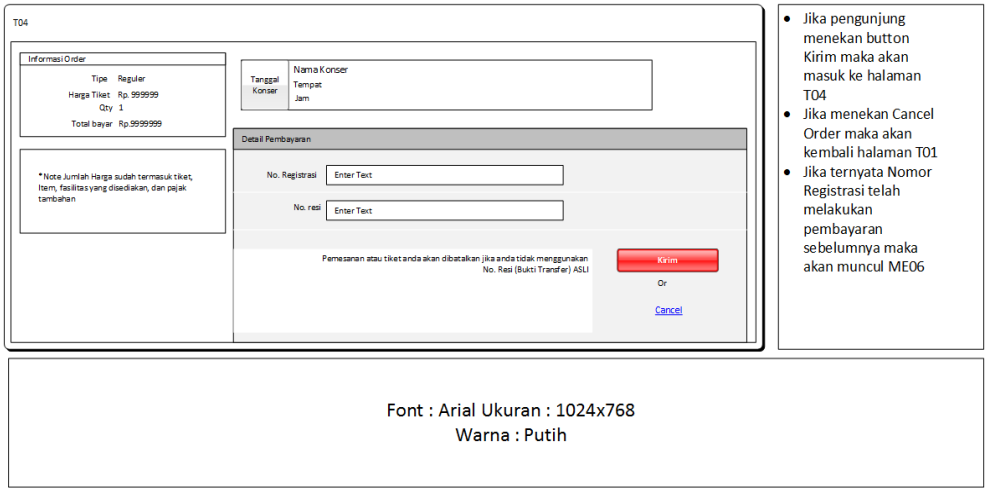
Halaman detail pemesanan merupakan halaman yang menampilkan detail pemesanan tiket kepada pengunjung. Adapun perancangan antarmuka untuk halaman detail pemesanan yang dibangun adalah seperti Gambar 3.58 berikut.



Gambar 3.58 Antarmuka Halaman Detail Pemesanan

3.8.4 Antarmuka Halaman Pembayaran (T04)

Halaman pembayaran merupakan halaman untuk pengunjung menginputkan nomor resi. Adapun perancangan antarmuka untuk halaman pembayaran yang dibangun adalah seperti Gambar 3.59 berikut.



Gambar 3.59 Antarmuka Halaman Pembayaran

3.8.5 Antarmuka Halaman Download Tiket (T05)

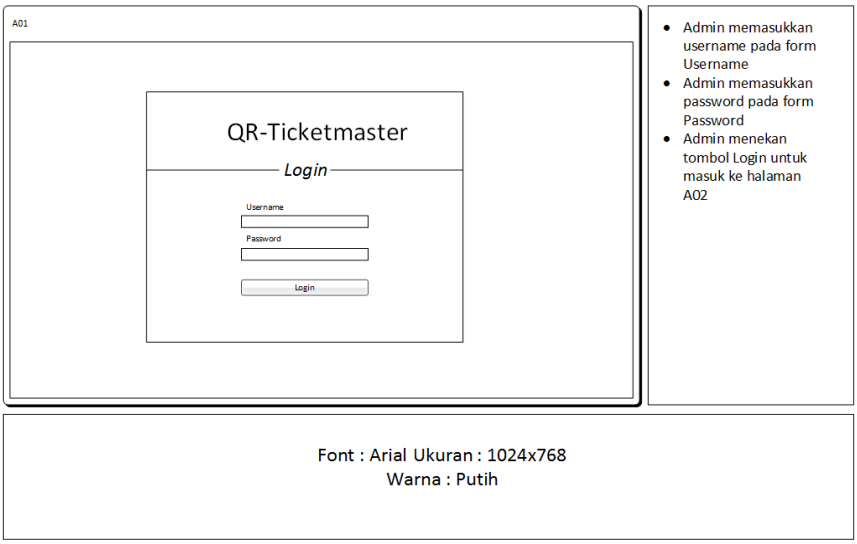
Halaman doenload tiket merupakan halaman yang menampilkan tiket yang dipesan pada pengunjung. Adapun perancangan antarmuka untuk halaman download tiket yang dibangun adalah seperti Gambar 3.60 berikut.



Gambar 3.60 Antarmuka Halaman Download Tiket

3.8.6 Antarmuka Halaman Login Admin (A01)

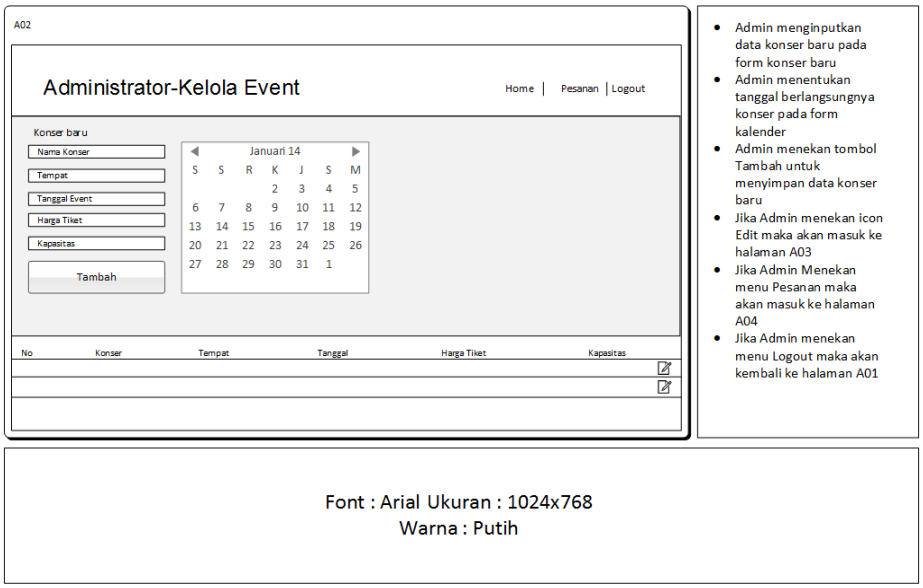
Halaman login admin merupakan halaman untuk login administrator. Adapun perancangan antarmuka untuk halaman login admin yang dibangun adalah seperti Gambar 3.61 berikut.



Gambar 3.61 Antarmuka Halaman Login Admin

3.8.7 Antarmuka Halaman Admin (A02)

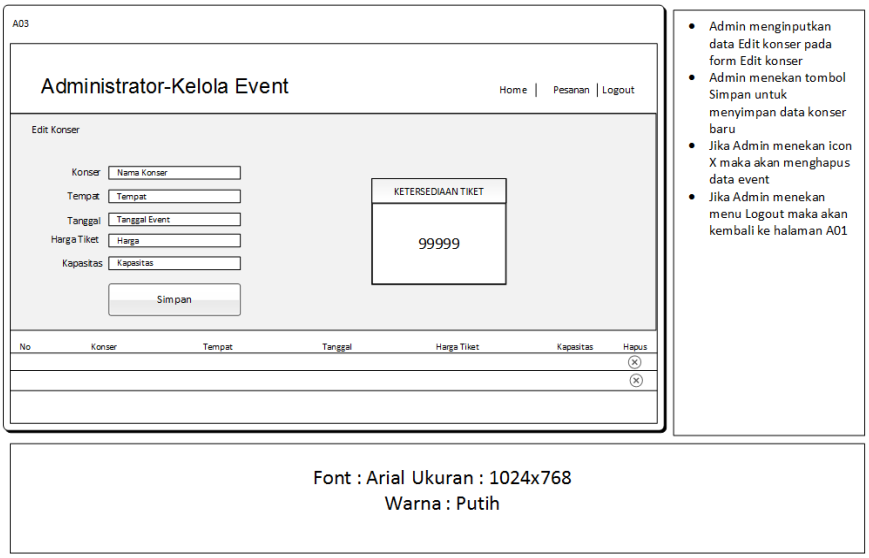
Halaman Admin merupakan halaman utama untuk administrator. Adapun perancangan antarmuka untuk halaman kelola *event* yang dibangun adalah seperti Gambar 3.62 berikut.



Gambar 3.62 Antarmuka Halaman Admin

3.8.8 Antarmuka Halaman Kelola Event (A03)

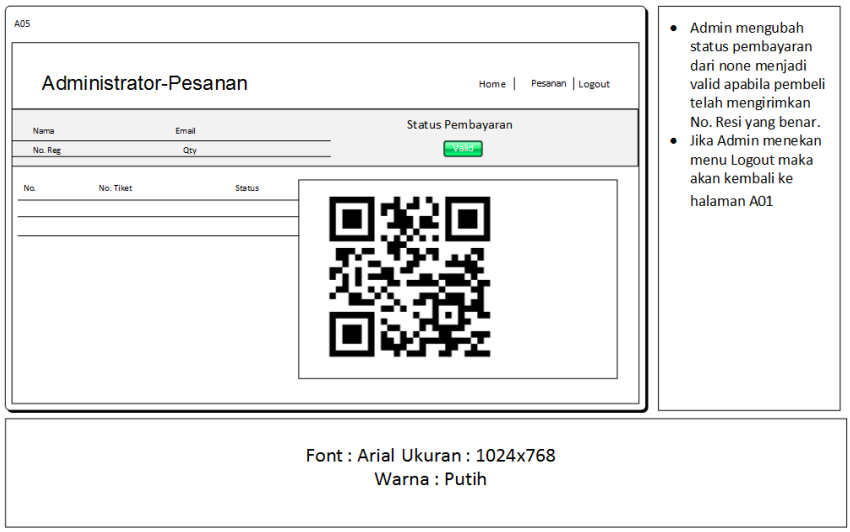
Halaman Kelola Event merupakan halaman untuk mengelola data event oleh administrator. Adapun perancangan antarmuka untuk halaman update *event* yang dibangun adalah seperti Gambar 3.63 berikut.



Gambar 3.63 Antarmuka Halaman Kelola Event

3.8.9 Antarmuka Halaman Pesanan (A04)

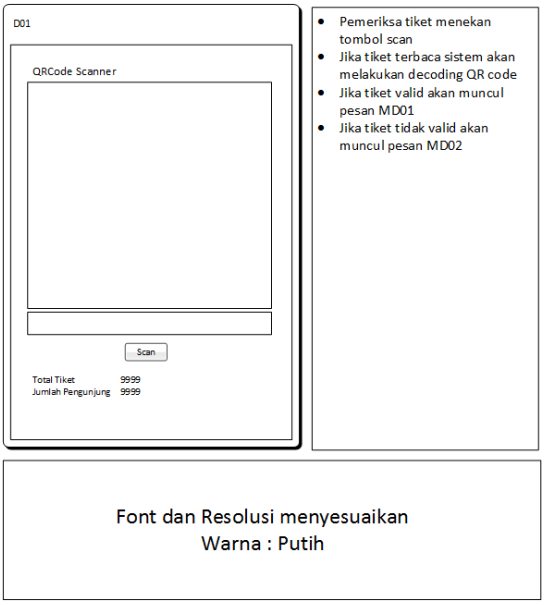
Halaman Pesanan merupakan halaman yang menampilkan data pesanan tiket. Adapun perancangan antarmuka untuk halaman pesanan yang dibangun adalah seperti Gambar 3.64 berikut.



Gambar 3.64 Antarmuka Halaman Pesanan

3.8.10 Antarmuka Decoder (D01)

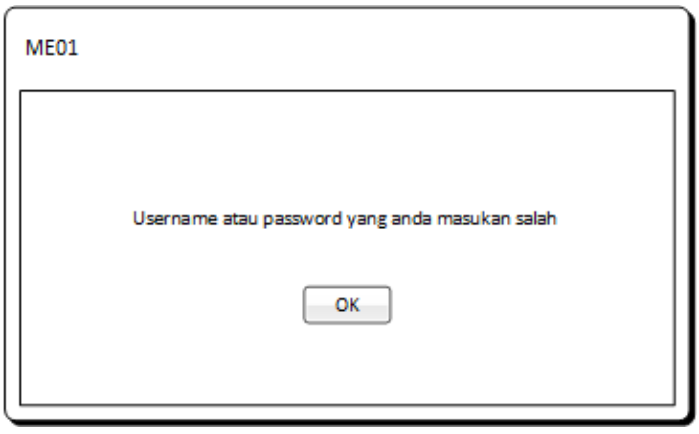
Decoder merupakan aplikasi mobile yang berfungsi sebagai *scanner* untuk verifikasi tiket. Adapun perancangan antarmuka untuk halaman *decoder* yang dibangun adalah seperti Gambar 3.65 berikut.



Gambar 3.65 Antarmuka Halaman Decoder

3.8.11 Antarmuka Pesan Login Gagal (ME01)

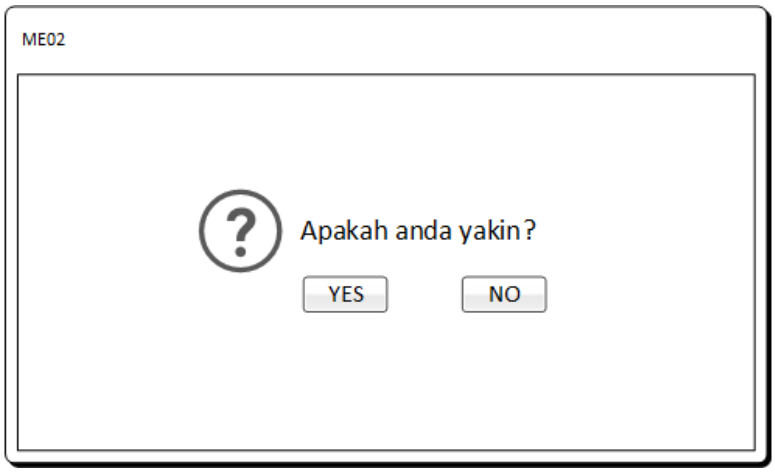
Pesan login gagal akan muncul bila administrator gagal login. Adapun perancangan antarmuka untuk pesan login gagal yang dibangun adalah seperti Gambar 3.66 berikut.



Gambar 3.66 Antarmuka Pesan Login Gagal

3.8.12 Antarmuka Pesan Konfirmasi Aksi (ME02)

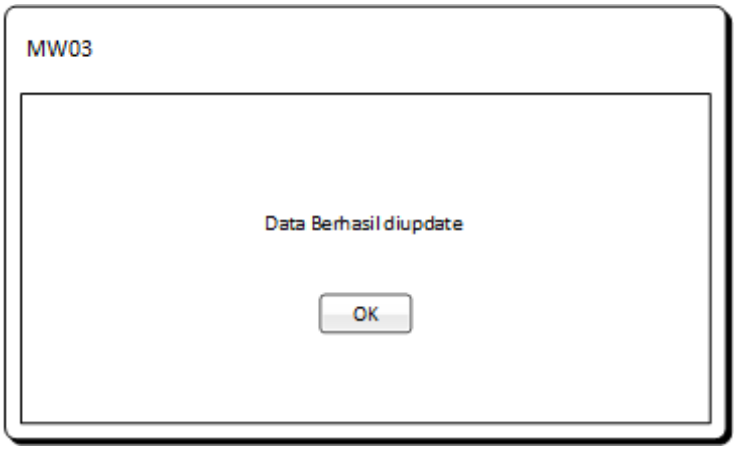
Pesan konfirmasi aksi akan muncul ketika sistem meminta konfirmasi aksi dari pengguna. Adapun perancangan antarmuka untuk pesan konfirmasi aksi yang dibangun adalah seperti Gambar 3.67 berikut.



Gambar 3.67 Antarmuka Pesan Konfirmasi Aksi

3.8.13 Antarmuka Pesan Tambah Data Berhasil (ME03)

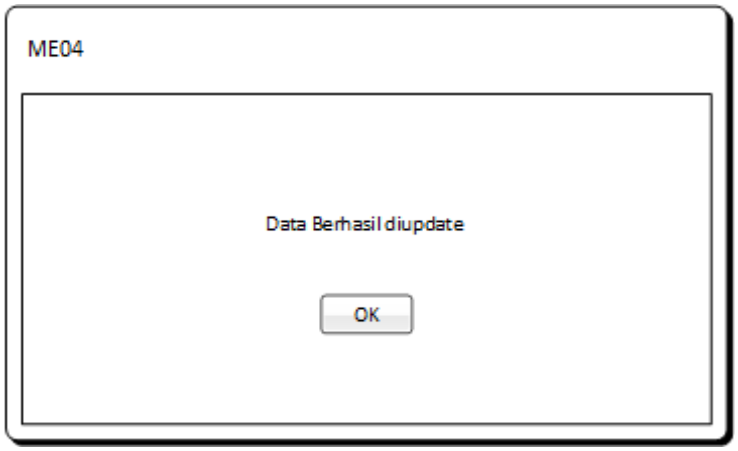
Pesan tambah data berhasil akan muncul apabila proses tambah data berhasil. Adapun perancangan antarmuka untuk pesan tambah data berhasil yang dibangun adalah seperti gambar 3.68 berikut



Gambar 3.68 Antarmuka Pesan Tambah Data Berhasil

3.8.14 Antarmuka Pesan Edit Data Berhasil (ME04)

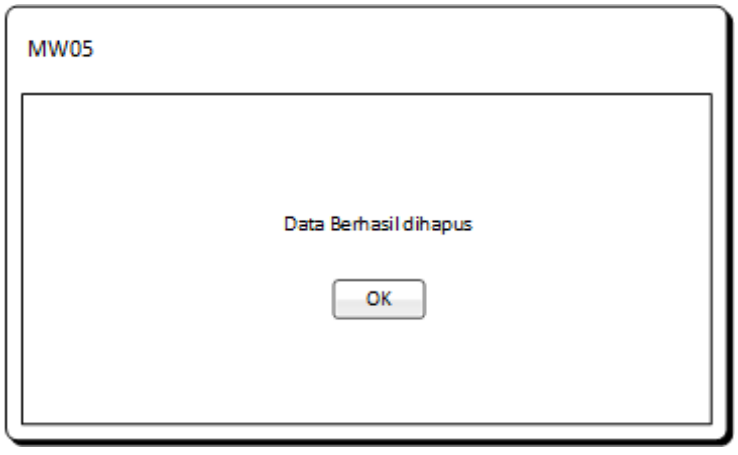
Pesan edit data berhasil akan muncul apabila proses edit berhasil. Adapun perancangan antarmuka untuk pesan edit data berhasil yang dibangun adalah seperti Gambar 3.69 berikut.



Gambar 3.69 Antarmuka Pesan Edit Data Berhasil

3.8.15 Antarmuka Pesan Hapus Data Berhasil (ME05)

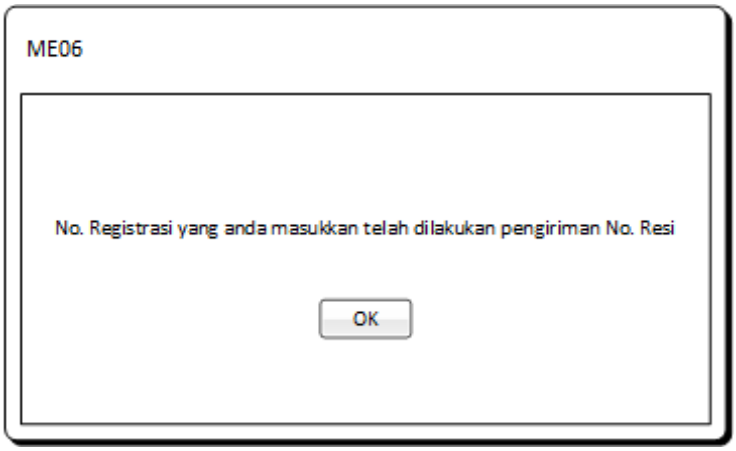
Pesan hapus data berhasil akan muncul apabila proses hapus berhasil Adapun perancangan antarmuka untuk pesan hapus data berhasil yang dibangun adalah seperti Gambar 3.70 berikut.



Gambar 3.70 Antarmuka Pesan Hapus Data Berhasil

3.8.16 Antarmuka Pesan Nomor Registrasi Salah (ME06)

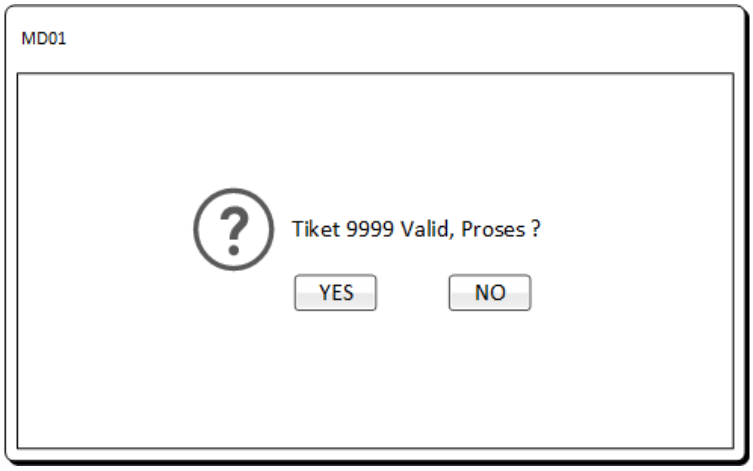
Pesan nomor registrasi salah akan muncul bila nomor registrasi yang diinputkan pengunjung salah. Adapun perancangan antarmuka untuk pesan nomor registrasi salah yang dibangun adalah seperti Gambar 3.71 berikut.



Gambar 3.71 Antarmuka Pesan Nomor Registrasi Salah

3.8.17 Antarmuka Pesan Konfirmasi Proses Tiket (MD01)

Pesan konfirmasi proses tiket akan muncul apabila tiket yang dipindai valid. Adapun perancangan antarmuka untuk pesan konfirmasi proses tiket yang dibangun adalah seperti Gambar 3.72 berikut.



Gambar 3.72 Antarmuka Pesan Konfirmasi Proses Tiket

3.8.18 Antarmuka Pesan Tiket Tidak Valid (MD02)

Pesan tiket tidak valid akan muncul apabila tiket yang dipindai tidak valid. Adapun perancangan antarmuka untuk pesan tiket tidak valid yang dibangun adalah seperti Gambar 3.73 berikut.



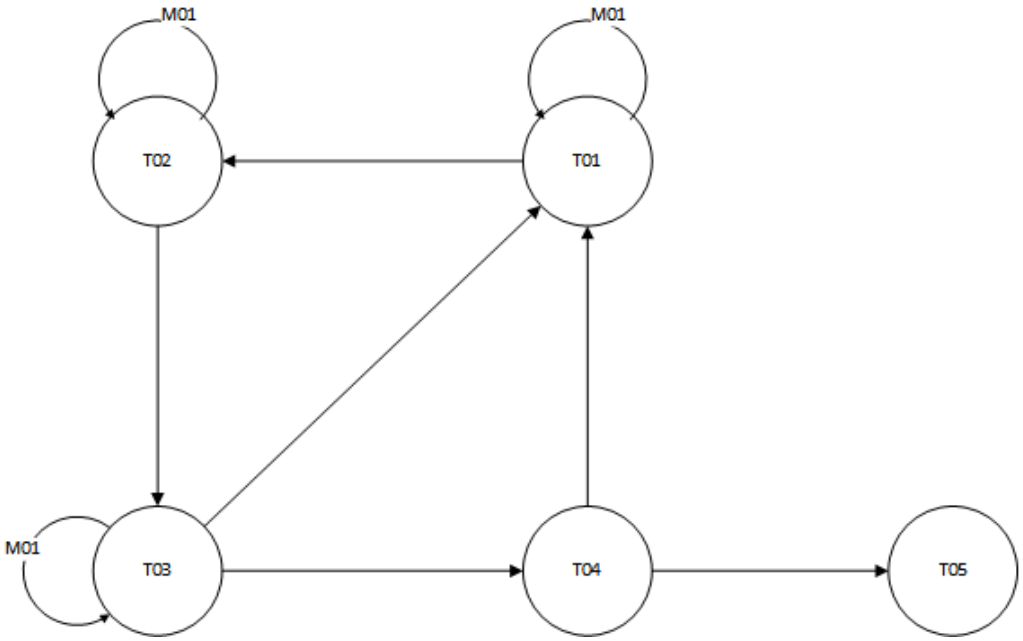
Gambar 3.73 Antarmuka Pesan Tiket Tidak Valid

3.9 Jaringan Semantik

Jaringan semantik merupakan gambaran pengetahuan grafis yang menunjukan hubungan yang saling berkaitan antar berbagai objek. Pada perancangan antarmuka ini, jaringan semantik terbagi menjadi dua berdasarkan jenis penggunanya.

3.9.1 Jaringan Semantik Halaman Pengunjung

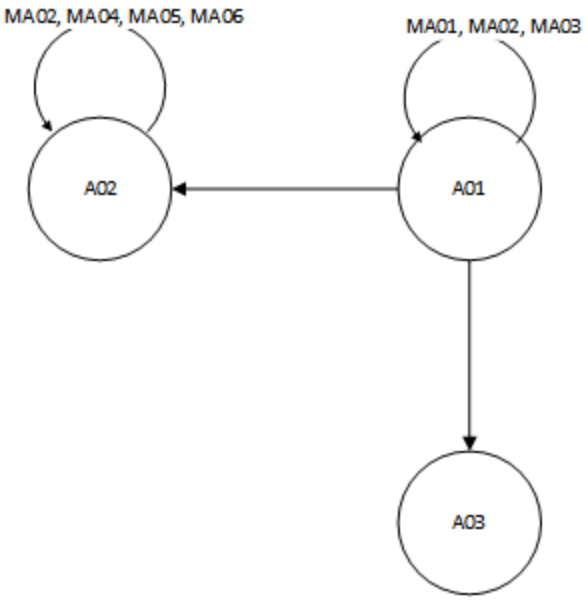
Halaman pengunjung merupakan halaman *website* yang dapat diakses oleh pengunjung. Gambar 3.74 merupakan jaringan semantik pada halaman pengunjung.



Gambar 3.74 Jaringan Semantik Halaman Pengunjung

3.9.2 Jaringan Semantik Halaman Admin

Halaman admin merupakan halaman website yang dapat diakses oleh Administrator. Gambar 3.75 merupakan jaringan semantik pada halaman admin.



Gambar 3.75 Jaringan Semantik Halaman Admin