BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah sistem MERISKA ini selesai dibangun maka dapat ditarik beberapa kesimpulan, yaitu:

1. Perangkat Lunak MERISKA telah berhasil dibangun sebagai aplikasi *customer relationship management* (CRM) yang mampu menangani pelayanan yang berbeda untuk setiap kelompok *customer* dengan metode pengelompokan *customer* berdasarkan *k-means clustering*.
2. Perangkat lunak MERISKA memberikan kemudahan untuk mengatur layanan yang berbeda untuk tiap kelompok *customer*.
3. Dari hasil pengujian dapat diketahui bahwa fungsi-fungsi yang disediakan oleh perangkat lunak MERISKA berjalan dengan benar dan sesuai dengan yang dikehendaki.
4. Pada pengujian perangkat lunak MERISKA ini didapatkan bahwa semakin besar data transaksi belanja *customer* maka memerlukan waktu yang lebih lama untuk melakukan pemrosesan data tetapi didapatkan suatu knowledge yang kuat untuk pengambilan suatu keputusan.

5.2 Saran

Penulis ingin memberikan beberapa saran untuk pengembangan lebih lanjut perangkat lunak MERISKA ini:

1. Penambahan fungsionalitas untuk layanan *customer* seperti forum, penawaran lewat sms, pemberian point hadiah untuk setiap pembelian buku,dll
2. Pembayaran yang dapat diintegrasikan dengan layanan pembyaran online seperti visa dan kartu kredit serta *online payment* lainnya.

DAFTAR PUSTAKA

Boggs Wendy, Michael Boggs, 2002, *Mastering UML with Rational Rose 2002*, SYBEX

Han Jiawei ,Kamber Micheline , 2001, *Data Mining : Concept and Techiques*, Morgan Kaufmann Publisher .

H.Dunham Margaret , 2003, *Data Mining : Introductory and Advanced Topics*, Pearson Education Inc , New Jersey.

Jalulaga, Anggisesa, 2007, *Pengembangan Aplikasi Customer Relationship Management Pada Perusahaan Asuransi Dengan Object Relational Database*, Universitas Atma Jaya Yogjakarta

Khera, Mandeep , 2000. *Customer Relationship Management – Beyond the "Buzz"*, United Customer Management Solution, ITToolBox Portal for CRM.

Harper, Simon, 2000, *Customer Relationship Management (CRM): well what is it really?* NHANZ Ltd., ITToolBox Portal for CRM

Findlay, Cliff, 2000, *CRM what's it all about*, CRM Consultant at Latitude Solutions Ltd, ITToolBox Portal for CRM

Sambasivan, TM., 2000, *CRM for Dummies*, ITToolBox Portal for CRM

Turban, Efraim et.all, 2001, *Information Technology for Management Third Edition*, John Willey & Sons Inc, New York

Turban, Efraim et.all, 2001, *Introduction to Information Technology*, John Willey & Sons Inc, New York

http://www.graphicophat.org/index.php?option=com_content&task=view&id=23, 14 Desember 2007

http://anakbinus.blogsome.com/2007/08/19/crm-customer-relation-management/, 14 Desember 2007

http://www.blogger.com/feeds/4975761385656584163/posts/default, 14 Desember 2007

http://en.wikipedia.org/wiki/Customer_relationship_managemet, 14 Desember 2007

http://sbinfocanada.about.com/cs/marketing/g/crm.htm, 14 Desember 2007

http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci213567,00.html, 14 Desember 2007

SKPL

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

PEMBANGUNAN APLIKASI *CUSTOMER RELATIONSHIP MANAGEMENT (CRM)* PADA TOKO BUKU ONLINE DENGAN KLASTERISASI (MERISKA)

Dipersiapkan oleh

Pherry Chandra 03.07.03782

Program Studi Teknik Informatika

Universitas Atma Jaya Yogyakarta

Jalan Babarsari 43 Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	Use Case Diagram: Hapus actor waktu, revisi use case login dan registrasi, revisi user case email notifikasi,revisi email penawaran buku
B	1. Revisi spesifikasi use case login, registrasi, email notifikasi, otomatisasi penawaran buku, pengelolaan klaster, pengelolaan order, order, pengisian keranjang belanja, pengelolaan katalog 2. Revisi analysis class diagram use use case login, registrasi, email notifikasi, otomatisasi penawaran buku, pengelolaan klaster, pengelolaan order, order, pengisian keranjang belanja, pengelolaan keranjang belanja, pengelolaan katalog 3. Revisi collaboration diagram untuk semua use case.
C	Entity Relationship Diagram (ERD)

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh	PHERRY							
Diperiksa oleh	PM ERN							
Disetujui oleh	PM ERN							

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi
9	A		
13-44	B		
47,48	C		

Daftar Isi

1 Pendahuluan 5
1.1 Tujuan 5
1.2 Lingkup Masalah 5
1.3 Definisi Akronim dan Singkatan 5
1.4 Referensi 6
1.5 Deskripsi umum (Overview) 6
2 Deskripsi Keseluruhan 8
2.1 Perspektif Produk 8
2.2 Kebutuhan Fungsionalitas Perangkat Lunak 11
2.2.1 Use Case: Login 11
2.2.2 Use Case: Pengubahan Profil Customer 12
2.2.3 Use Case: Pencarian Catalog 12
2.2.4 Use Case: Pengisian Keranjang Belanja 12
2.2.5 Use Case: Order 12
2.2.6 Use Case: Pengelolaan Catalog 12
2.2.7 Use Case: Pengelolaan Keranjang Belanja 13
2.2.8 Use Case: Pengelolaan Order 13
2.2.9 Use Case: Pengelolaan Klaster 13
2.2.10 Use Case: Notifikasi Pembayaran 13
2.2.11 Use Case: Tampil Informasi 14
2.2.12 Use Case: Pengelolaan Email 14
2.2.13 Use Case: Pengelolaan Stok 14
2.3 Karakteristik Pengguna 14
2.3.1 Administrator 14
2.3.2 Customer 14
3 Deskripsi Rinci Kebutuhan 15
3.1 Spesifikasi Kebutuhan Fungsionalitas 15
3.1.1 Spesifikasi Use Case 15
3.1.2 Static Structure Diagram 29
3.1.3 Interaction Diagram 36
3.2 Spesifikasi Kebutuhan Non-Fungsionalitas 47
3.2.1 Kebutuhan Antarmuka Eksternal 47
3.2.2 Antarmuka Pemakai 47
3.2.3 Antarmuka Perangkat Keras 47
3.2.4 Antarmuka Perangkat Lunak 47
3.2.5 Antarmuka Komunikasi 48
3.3 Spesifikasi Kebutuhan Data 48
3.3.1 ERD (Entity Relationship Diagram) 49

1 Pendahuluan

1.1 Tujuan

Dokumen SKPL ini dibuat untuk menyediakan deskripsi lengkap mengenai kebutuhan fungsionalitas dan kebutuhan non-fungsionalitas dari perangkat lunak *customer relationship management klaster* (MERISKA). Dokumen ini ditujukan untuk pembuat *e-commerce system* yang berfokus kepada kebutuhan *customer*. Pembangunan perangkat lunak ini diimplementasikan pada toko buku *online*. *Customized system* yang dibangun menggunakan algoritma *k-means clustering*.

1.2 Lingkup Masalah

Perangkat lunak MERISKA dikembangkan dengan tujuan untuk :

1. Menangani proses belanja secara *online*.
2. Menangani klasterisasi.
3. Menangani pengelolaan tampilan webpage yang sesuai dengan karakteristik klaster.
4. Menangani pengelolaan diskon yang berbeda untuk setiap klaster.
5. Menangani pengelolaan informasi yang berbeda untuk setiap klaster.
6. Menangani otomatisasi penawaran produk yang berbeda untuk setiap klaster.

1.3 Definisi Akronim dan Singkatan

Daftar definisi dan akronim yang digunakan :

Keyword atau Phrase	Definisi
Hak Akses	**Hak yang dimiliki oleh user untuk menggunakan sistem. Hak akses diatur oleh Administrator**
MERISKA	**Customer Customized System**
Server	**Komputer yang menyediakan sumber daya bagi klien yang**

	terhubung melalui jaringan
Database	**Kumpulan data yang terkait yang diorganisasikan dalam struktur tertentu dan dapat diakses dengan cepat.**
Role	**Hak yang dimiliki oleh user untuk menggunakan sitem. Hak akses diatur oleh Administrator**
UML	**bahasa (notasi) pemodelan perangkat lunak berorientasi obyek**
User Interface Information	**Informasi mengenai antarmuka pemakai dengan system**
SKPL	**Dokumen yang berisi tentang spesifikasi kebutuhan pengembagna perangkat lunak.**
LAN	**Local Area Network atau jaringan lokal komputer**
DBMS	**DataBase Management System atau pengelola manajemen data base**

Tabel 1.1 Tabel definisi akronim dan singkatan

1.4 Referensi

Dokumen yang digunakan sebagai acuan dalam rencana pengembangan perangkat lunak ini adalah :

- Wendy Boggs, Michael Boggs. *Mastering UML With Rational Rose.* 2002.

1.5 Deskripsi umum (Overview)

Dokumen SKPL ini dibagi menjadi tiga bab. Bab pertama adalah **Pendahuluan,** yang berisi tentang deskripsi dokumen. Bab kedua adalah **Deskripsi Keseluruhan,** yang berisi penjelasan secara umum mengenai sistem yang akan dikembangkan meliputi fungsi-fungsi dari sistem, karakteristik pengguna, batasan dan asumsi yang diambil dalam pengembangan perangkat lunak. Bab ketiga adalah **Spesifikasi Rinci Kebutuhan,** yang berisi penjelasan tentang kebutuhan sistem yang akan dikembangkan secara lebih rinci dan **Realisasi Use Case**, yang berisi

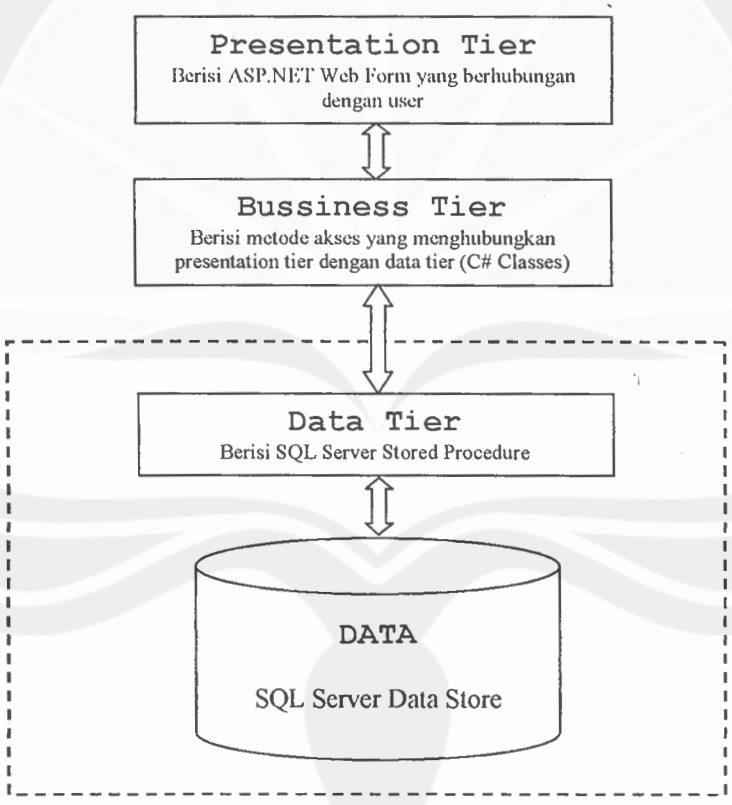
realisasi use case dalam tahap analisis (konseptual), yang akan digunakan sebagai dasar realisasi use case pada tahap desain.

2 Deskripsi Keseluruhan

2.1 Perspektif Produk

Pembangunan *Customer Relationship Management Klaster (MERISKA)* ini ditujukan pada pengembangan sistem yang berdasarkan model *customer* yang telah terbentuk dari proses klasterisasi terhadap data belanja *customer*. Pembangunan Aplikasi *Customer Relationship Management Klaster (MERISKA)* pada toko buku *online* ini berkaitan dengan adanya proses penerapan *MERISKA* pada aktivitas *e-commerce* toko buku *online* tersebut.

Aplikasi ini akan dibangun dengan mengunakan arsitektur 3 lapisan (*Three Tier Architecture*) yang terpisah yaitu:



Gambar 2.1 Three Tier Architecture *MERISKA*

Proses-proses dalam sistem antara lain:

1. Klasterisasi model *customer* secara otomatis.

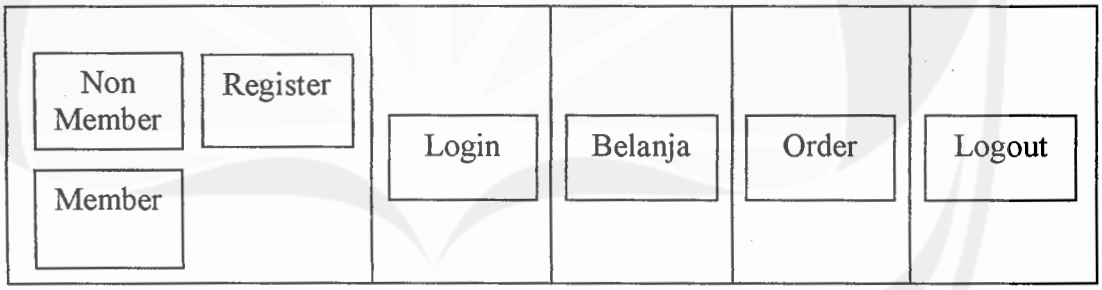
Model *customer* dibentuk berdasarkan klasterisasi terhadap data belanja *customer*. Proses klasterisasi ini dilakukan pada lapisan *Data Tier*, hal ini dilakukan agar akses data lebih cepat. Dengan menggunakan fasilitas dari *SQL Server Job*, proses klasterisasi dapat diatur (*schedule*) secara otomatis.

2. Otomatisasi penawaran produk baru melalui e-mail kepada *customer* berdasarkan model klasterisasi yang telah terbentuk.

   Otomatisasi penawaran produk baru melalui e-mail juga dilakukan pada lapisan *Data Tier*. Otomatisasi dilakukan dengan menggunakan fasilitas dari *SQL Server Job* dan *Database Mail* dari SQL Server 2005.

3. *Shopping pipeline* (track proses belanja).

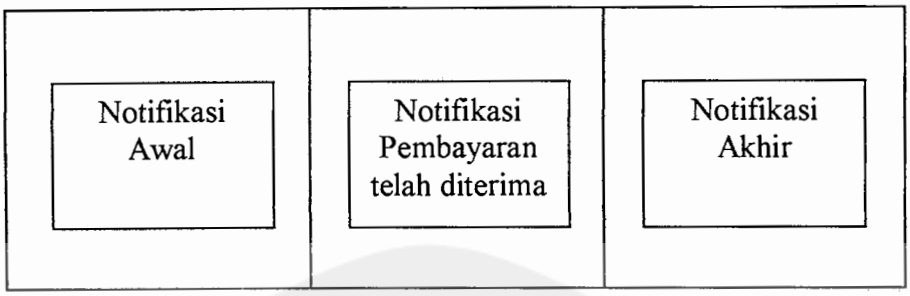
   Shopping Pipeline merupakan suatu track proses yang menuntun *customer* dalam melakukan belanja secara *online*. *Shopping Pipeline* diimplementasikan pada lapisan *Presentation Tier*.



Gambar 2.2 ***Shopping Pipeline***

4. *Order pipeline* (track proses order).

   Order Pipeline merupakan suatu track proses order yang terjadi setelah *customer* melakukan order. Implementasi *Order Pipeline* dilakukan pada lapisan *Presentation Tier*.

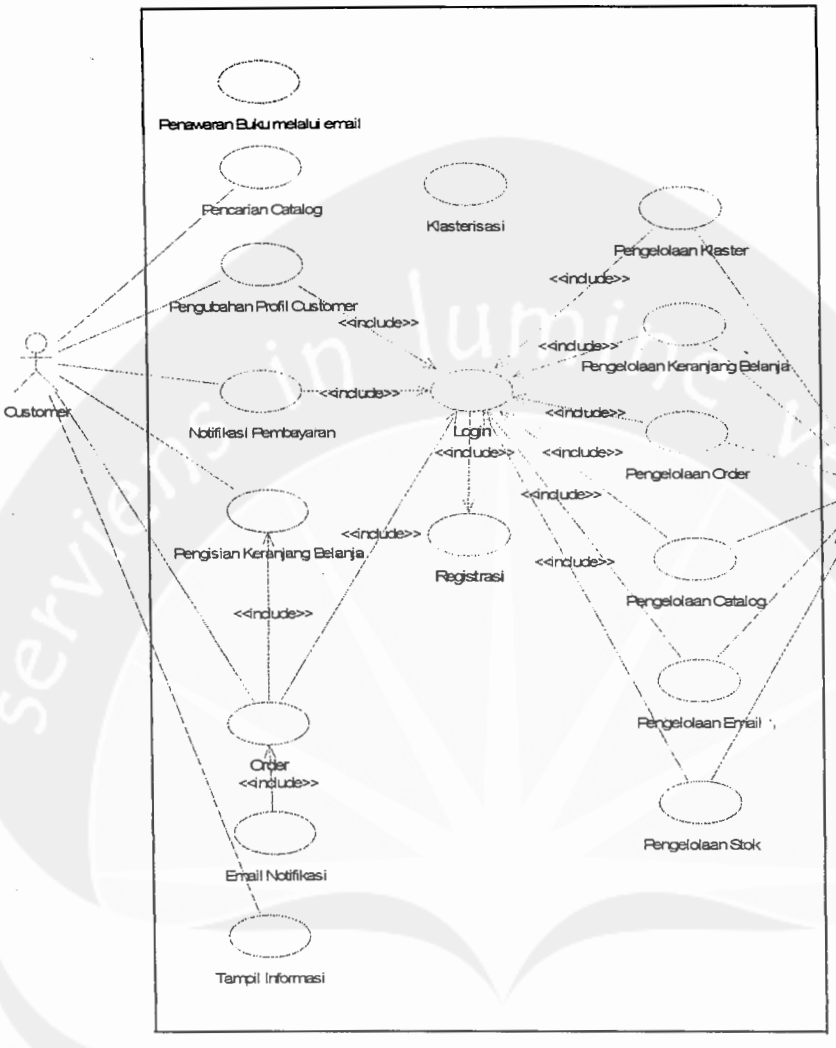


Gambar 2.3 ***Order Pipeline***

- Notifikasi Awal: notifikasi email kepada *customer* bahwa order telah diterima oleh toko.
- Pembayaran: notifikasi email kepada *customer* bahwa pembayaran telah diterima oleh toko dan barang akan segera dikirimkan ke *customer*.
- Notifikasi Akhir: notifikasi email kepada *customer* bahwa barang telah dikirim dan ucapan terima kasih telah berbelanja.

5. Menampilkan *content page* berdasarkan model klasterisasi *customer* yang telah dibangun (penawaran produk lama dan baru yang sejenis dan penawaran produk sejenis yang paling sering dibeli).
6. Management diskon dan informasi tiap klaster

2.2 Kebutuhan Fungsionalitas Perangkat Lunak



Gambar 2.4 Use Case Diagram

2.2.1 Use Case: Login

Use Case ini digunakan oleh aktor baik *administrator* maupun *customer* untuk memperoleh akses ke sistem. *Login* didasarkan pada *username* masing-masing aktor yang unik dan password.

Lihat: **Spesifikasi Use Case: Login (UC- MERISKA -01).**

2.2.2 Use Case: Pengubahan Profil Customer

Use case ini memberi akses kepada aktor *customer* untuk mengubah *profile customer* yang telah tersimpan ketika use case registrasi telah dijalankan.

Lihat: **Spesifikasi Use Case: Pengubahan Profil Customer (UC-MERISKA -02).**

2.2.3 Use Case: Pencarian Catalog

Use case ini memberi akses kepada aktor *customer* untuk mencari buku-buku yang tersimpan didalam katalog sistem.

Lihat: **Spesifikasi Use Case: Pencarian Catalog (UC- MERISKA -03).**

2.2.4 Use Case: Pengisian Keranjang Belanja

Use case ini merupakan bentuk keranjang belanja abstrak yang berfungsi untuk mengumpulkan buku-buku yang akan dibeli sebelum dilakukan proses order terhadap buku-buku tersebut. Pada use case ini terhadap fungsi untuk meng-update jumlah buku dan menghapus data buku dari keranjang belanja.

Lihat: **Spesifikasi Use Case: Pengisian Keranjang Belanja (UC-MERISKA -04).**

2.2.5 Use Case: Order

Use case ini merupakan proses order buku yang dilakukan secara online, mengikuti proses *order pipeline* sistem.

Lihat: **Spesifikasi Use Case: Pembelian (UC- MERISKA -05).**

2.2.6 Use Case: Pengelolaan Catalog

Use case ini digunakan oleh *administrator* untuk mengelola katalog buku yang meliputi penambahan buku baru, pengubahan buku dan penghapusan buku.

Lihat: **Spesifikasi Use Case: Pengelolaan Catalog (UC- MERISKA -06).**

2.2.7 Use Case: Pengelolaan Keranjang Belanja

Use case ini memberi akses kepada *administrator* untuk mengorganisasi data keranjang belanja yang tersimpan di database. Pengorganisasian disini yaitu menghapus data keranjang belanja yang sudah tidak diperlukan lagi.

Lihat: **Spesifikasi Use Case: Pengelolaan Keranjang Belanja (UC-MERISKA -07).**

2.2.8 Use Case: Pengelolaan Order

Use case ini memberi akses kepada *administrator* untuk mengorganisasi data order buku yang tersimpan di database. Pengorganisasian disini yaitu mengatur status order.

Lihat: **Spesifikasi Use Case: Pengelolaan Order (UC- MERISKA -08).**

2.2.9 Use Case: Pengelolaan Klaster

Use case ini digunakan oleh *administrator* untuk memanagement event terhadap klaster, mengedit label klaster serta melihat anggota-anggota dalam klaster dan karakteristik klaster. Untuk proses klasterisasinya sendiri dilakukan oleh use case klasterisasi.

Lihat: **Spesifikasi Use Case: Pengelolaan Klaster (UC- MERISKA -09).**

2.2.10 Use Case: Notifikasi Pembayaran

Use case ini digunakan oleh *customer* untuk memberitahukan kepada sistem bahwa proses pembayaran telah dilakukan kemudian sistem akan menyampaikan informasi tersebut kepada *administrator* dalam bentuk email notifikasi pembayaran.

Lihat: **Spesifikasi Use Case: Notifikasi Pembayaran (UC- MERISKA -10).**

2.2.11 Use Case: Tampil Informasi

Use case ini digunakan oleh *customer* untuk melihat informasi-infromasi yang disediakan oleh sistem seperti informasi katalog buku dan informasi buku detail.

Lihat: **Spesifikasi Use Case: Tampil Informasi (UC- MERISKA -11).**

2.2.12 Use Case: Pengelolaan Email

Use case ini digunakan oleh *administrator* untuk mengirim email-email notifikasi ke customer berdasarkan proses order yang terjadi.

Lihat: **Spesifikasi Use Case: Pengelolaan Email (UC- MERISKA -12).**

2.2.13 Use Case: Pengelolaan Stok

Use case ini digunakan oleh *administrator* untuk mengatur stok buku dalam katalog.

Lihat: **Spesifikasi Use Case: Pengelolaan Stok (UC- MERISKA -13).**

2.3 Karakteristik Pengguna

2.3.1 Administrator

a. Memahami pengoperasian komputer.
b. Familiar dengan *internet* (*world wide web*).
c. Memahami basis data dengan SQL Server 2005.
d. Memahami aplikasi MERISKA toko buku *online.*
e. Mengerti tentang penggunaan email.

2.3.2 Customer

a. Memahami pengoperasian komputer.
b. Mampu menggunakan aplikasi MERISKA toko buku *online*.
c. Mengerti proses belanja *online.*
d. Familiar dengan *internet* (*world wide web*).
e. Mengerti tentang penggunaan email.

3 Deskripsi Rinci Kebutuhan

3.1 Spesifikasi Kebutuhan Fungsionalitas

3.1.1 Spesifikasi Use Case

3.1.1.1 Spesifikasi Use Case: Login

Tabel 3.1 Spesifikasi Use Case: Login

Use Case ID	UC- MERISKA -01
Use Case Name	Login
Use Case Type	-
Priority	-
Actors	Customer, Administrator
Description	Use Case ini digunakan untuk memperoleh akses ke sistem. *Login* didasarkan pada *username* masing-masing aktor yang unik dan password.
Preconditions	Aktor telah mempunyai *username* untuk akses ke sistem.
Basic Path	1. Sistem menampilkan antarmuka untuk login. 2. Aktor memasukkan username dan password. 3. Sistem memeriksa username dan password aktor. 4. Sistem memberikan akses ke aktor sesuai Role-nya (Administrator/ Customer).
Alternative Path	**Registrasi** 1. Sistem menampilkan antarmuka untuk registrasi customer. 2. Aktor memasukkan data-data (username dan profil customer) yang diminta. 3. Sistem memeriksa data-data (username dan profil customer) yang diisikan aktor. 4. Sistem menyimpan data-data tersebut ke basis data.
Postconditions	Aktor mendapat akses untuk menggunakan fungsi-fungsi pada sistem.
Exception Paths	1. Username atau password tidak sesuai (setelah Tabel 3.1 Basic Path 3) a. Sistem menampilkan peringatan bahwa username atau password tidak sesuai.

	b. Kembali ke Tabel 3.1 Basic path 2. 2. Username tidak terdapat dalam database (setelah Tabel 3.1 Basic Path 3) a. Sistem menampilkan peringatan bahwa username tidak terdapat di database. b. Kembali ke Tabel 3.1 Basic path 2. 3. Data-data yang diisikan tidak lengkap (setelah Tabel 3.1 Alternative Path 3) a. Sistem menampilkan peringatan bahwa data yang diisikan belum lengkap. b. Kembali ke Tabel 3.1 Alternative Path 2. 4. Username yang diisikan sudah digunakan oleh orang lain (setelah Tabel 3.1 Alternative Path 3) a. Sistem menampilkan peringatan bahwa username yang diisikan sudah digunakan, aktor diminta untuk menggantinya. b. Kembali ke Tabel 3.1 Alternative Path 2. 5. Penulisan format email tidak benar (setelah Tabel 3.1 Alternative Path 3) a. Sistem menampilkan peringatan bahwa penulisan format email tidak benar, aktor diminta untuk menggantinya. b. Kembali ke Tabel 3.1 Alternative Path 2.
Extends	-
Includes	-

3.1.1.2 Spesifikasi Use Case: Pengubahan Profil Customer

Tabel 3.2 Spesifikasi Use Case: Pengubahan Profil Customer

Use Case ID	UC- MERISKA -02
Use Case Name	Pengubahan Profil Customer
Use Case Type	-
Priority	-
Actors	Customer
Description	Use case ini digunakan untuk mengubah *profile customer* yang telah tersimpan.
Preconditions	Use Case : Login sudah dilaksanakan dan aktor sudah berhasil

	memasuki sistem.
Basic Path	1. Sistem menampilkan antarmuka untuk pengubahan data profil *customer*, dimana data profil customer yang sudah ada langsung ditampilkan. 2. Aktor mengaktifkan form edit data profil.(button edit) 3. Aktor mengubah data profil pada tempat yang disediakan. 4. Aktor meminta sistem untuk menyimpan perubahan dengan memicu aksi update. 5. Sistem memeriksa data-data yang bersangkutan. 6. Sistem menyimpan data ke basis data.
Alternative Path	-
Postconditions	Data customer telah ter-update
Exception Paths	1. Data-data yang diisikan tidak lengkap (setelah Tabel 3.3 Basic Path 5) a. Sistem menampilkan peringatan bahwa data yang diisikan belum lengkap. b. Kembali ke Tabel 3.3 Basic Path 3. 2. Penulisan format email tidak benar (setelah Tabel 3.2 Basic Path 5) a. Sistem menampilkan peringatan bahwa penulisan format email tidak benar, aktor diminta untuk menggantinya. b. Kembali ke Tabel 3.2 Basic Path 3.
Extends	-
Includes	-Use Case: Login

3.1.1.3 Spesifikasi Use Case: Pencarian Catalog

Tabel 3.3 Spesifikasi Use Case: Pencarian Catalog

Use Case ID	UC- MERISKA -03
Use Case Name	Pencarian Catalog
Use Case Type	-
Priority	-
Actors	Customer
Description	Use case ini digunakan untuk mencari buku-buku yang tersimpan didalam katalog sistem.
Preconditions	-

Basic Path	1. Sistem menampilkan antarmuka untuk mencari dan menampilkan data-data buku. 2. Aktor memasukkan atribut pencarian data buku. 3. Aktor meminta sistem untuk mencari data buku berdasarkan atribut yang dimasukkan dengan memicu aksi cari. 4. Sistem memeriksa data buku yang dicari di basis data. 5. Sistem menampilkan data-data buku bersangkutan.
Alternative Path	-
Postconditions	Hasil pencarian buku ditampilkan.
Exception Paths	1. Atribut pencarian masih kosong (setelah Tabel 3.3 Basic Path 4) a. Sistem menampilkan peringatan bahwa atribut data pencarian masih kosong. b. Kembali ke Tabel 3.3 Basic Path 2. 2. Data buku yang dicari tidak ada (setelah Tabel 3.3 Basic Path 4) a. Sistem menampilkan peringatan bahwa query result pencarian kosong. b. Kembali ke Tabel 3.3 Basic Path 2.
Extends	-
Includes	-

3.1.1.4 Spesifikasi Use Case: Pengisian Keranjang Belanja

Tabel 3.4 Spesifikasi Use Case: Pengisian Keranjang Belanja

Use Case ID	UC- MERISKA -04
Use Case Name	Pengisian Keranjang Belanja
Use Case Type	-
Priority	-
Actors	Customer
Description	Use case ini merupakan bentuk keranjang belanja abstrak yang berfungsi untuk mengumpulkan buku-buku yang akan dibeli sebelum dilakukan proses order terhadap buku-buku tersebut.
Preconditions	-
Basic Path	**Tambah data buku ke keranjang belanja**

	1. Sistem menampilkan antarmuka yang berisi data-data buku / catalog buku yang dapat dimasukan ke dalam keranjang belanja. 2. Aktor memasukan data buku yang akan diorder ke dalam keranjang belanja. 3. Jika aktor masih ingin melanjutkan belanja maka kembali ke Tabel 3.4 Basic Path 2, jika tidak maka ke Tabel 3.4 Basic Path 4. 4. Aktor melakukan checkout. 5. Sistem menyimpan data keranjang belanja ke basis data.
Alternative Path	**A-1. Ubah jumlah buku** 1. Sistem menampilkan antarmuka yang berisi fungsi untuk meng-ubah jumlah setiap buku yang berada di keranjang belanja. 2. Aktor melakukan perubahan jumlah buku terhadap data buku yang ingin diupdate jumlahnya. 3. Aktor meminta sistem untuk mengubah jumlah buku bersangkutan dalam keranjang belanja dengan memicu aksi update. 4. Sistem memeriksa ke-valid-an jumlah buku. 5. Sistem mengubah jumlah buku bersangkutan dalam keranjang belanja. **A-2.Hapus data buku dari keranjang belanja** 1. Sistem menampilkan antarmuka yang berisi fungsi untuk meng-hapus data buku yang berada di keranjang belanja. 2. Aktor memilih data buku yang ingin dihapus dalam keranjang belanja. 3. Aktor meminta sistem untuk menghapus data buku yang telah dipilih dengan memicu aksi hapus. 4. Sistem menghapus data buku dari keranjang belanja.
Postconditions	Data-data buku dalam keranjang belanja sudah siap untuk diorder.
Exception Paths	1. Jumlah buku yang dimasukan tidak valid (setelah Tabel 3.4 Alternative Path A-1.4) a. Sistem menampilkan peringatan bahwa proses update jumlah buku gagal. b. Kembali ke Tabel 3.4 Alternative Path A-1.2.

	2. Jumlah buku yang dimasukan nol (setelah Tabel 3.4 Alternative Path A-1.4) a. Sistem akan melakukan proses hapus data buku yang bersangkutan.
Extends	-
Includes	-

3.1.1.5 Spesifikasi Use Case: Order

Tabel 3.5 Spesifikasi Use Case: Order

Use Case ID	UC- MERISKA -05
Use Case Name	Order
Use Case Type	-
Priority	-
Actors	Customer
Description	Use case ini merupakan proses order buku yang dilakukan secara online.
Preconditions	Use Case: Keranjang Belanja dan Use Case: Login telah dilakukan.
Basic Path	1. Sistem menampilkan antarmuka yang berisi data-data buku yang akan dibeli serta jumlah total harga dan data customer yang bersangkutan sebagai konfirmasi awal sebelum proses order pembelian dilakukan. 2. Aktor meminta sistem untuk melakukan proses order pembelian dengan memicu aksi order. 3. Sistem mengaktifkan proses order pembelian.
Alternative Path	-
Postconditions	Proses order pembelian telah diaktifkan oleh sistem.
Exception Paths	-
Extends	-
Includes	1. Use Case: Pengisian Keranjang Belanja 2. Use Case: Login

3.1.1.6 Spesifikasi Use Case: Pengelolaan Catalog

Tabel 3.6 Spesifikasi Use Case: Pengelolaan Catalog

Use Case ID	UC- MERISKA -06

Use Case Name	Pengelolaan Catalog
Use Case Type	-
Priority	-
Actors	Administrator
Description	Use case ini digunakan untuk mengelola katalog buku yang meliputi penambahan buku baru, pengubahan buku dan penghapusan buku.
Preconditions	Use Case: Login sudah dilaksanakan dan aktor sudah berhasil memasuki sistem.
Basic Path	**Tambah Kategori** 1. Sistem menampilkan antarmuka untuk tambah kategori. 2. Aktor memasukan kategori baru. 3. Aktor meminta sistem untuk menyimpan kategori baru tersebut dengan memicu aksi tambah kategori. 4. Sistem memeriksa kategori baru tersebut. 5. Sistem menyimpan kategori baru tersebut ke basis data.
Alternative Path	**A-1.Tambah Data Buku Baru** 1. Sistem menampilkan antarmuka untuk tambah data buku baru. 2. Aktor memasukan data buku. 3. Aktor meminta sistem untuk menyimpan data buku baru tersebut dengan memicu aksi tambah buku. 4. Sistem memeriksa data buku baru tersebut. 5. Sistem menyimpan data buku baru tersebut ke basis data. **A-2.Ubah Kategori** 1. Sistem menampilkan antarmuka yang berisi kategori. 2. Aktor memilih kategori yang akan diubah. 3. Aktor melakukan perubahan terhadap kategori tersebut. 4. Aktor meminta sistem untuk melakukan perubahan terhadap kategori dengan memicu aksi ubah. 5. Sistem memeriksa kategori yang bersangkutan. 6. Sistem menyimpan kategori bersangkutan ke basis data. **A-3.Ubah data buku** 1. Sistem menampilkan antarmuka yang berisi data-data buku.

	2. Aktor memilih data buku yang akan diedit. 3. Aktor melakukan perubahan terhadap data buku. 4. Aktor meminta sistem untuk melakukan perubahan terhadap data buku dengan memicu aksi ubah. 5. Sistem memeriksa data buku yang bersangkutan. 6. Sistem menyimpan data buku ke basis data. **A-4.Hapus Kategori** 1. Sistem menampilkan antarmuka yang berisi fungsi untuk meng-hapus kategori. 2. Aktor memilih kategori yang ingin dihapus. 3. Aktor meminta sistem untuk menghapus kategori yang telah dipilih dengan memicu aksi hapus. 4. Sistem menghapus kategori dari basis data. **A-5.Hapus data buku** 1. Sistem menampilkan antarmuka yang berisi fungsi untuk meng-hapus data buku. 2. Aktor memilih data buku yang ingin dihapus. 3. Aktor meminta sistem untuk menghapus data buku yang telah dipilih dengan memicu aksi hapus. 4. Sistem menghapus data buku dari basis data.
Postconditions	Data Katalog di basis data ter-update
Exception Paths	1. Detail data buku yang diisikan belum lengkap (setelah Tabel 3.6 Alternative Path A-1.4) a. Sistem menampilkan peringatan bahwa data buku yang diisikan belum lengkap. b. Kembali ke Tabel 3.6 Alternative Path A-1.2. 2. Data buku yang dimasukan sudah ada di basis data. (setelah Tabel 3.6 Alternative Path A-1.4) a. Sistem menampilkan peringatan bahwa data buku sudah ada di basis data. b. Kembali ke Tabel 3.6 Alternative Path A-1.2. 3. Detail data buku yang akan diupdate belum lengkap (setelah Tabel 3.6 Alternative Path A-3.5) a. Sistem menampilkan peringatan bahwa data buku yang akan diubah belum lengkap. b. Kembali ke Tabel 3.6 Alternative Path A-3.3. 4. Kategori yang dimasukan sudah ada di basis data. (setelah

	Tabel 3.6 Basic Path 4) a. Sistem menampilkan peringatan bahwa kategori sudah ada di basis data. b. Kembali ke Tabel 3.6 Basic Path 2.
Extends	-
Includes	-Use Case: Login

3.1.1.7 Spesifikasi Use Case: Pengelolaan Keranjang Belanja

Tabel 3.7 Spesifikasi Use Case: Pengelolaan Keranjang Belanja

Use Case ID	UC- MERISKA -07
Use Case Name	Pengelolaan Keranjang Belanja
Use Case Type	-
Priority	-
Actors	Administrator
Description	Use case ini digunakan untuk mengorganisasi data keranjang belanja yang tersimpan di database. Pengorganisasian disini yaitu menghapus data keranjang belanja yang sudah tidak diperlukan lagi.
Preconditions	Use Case: Login sudah dilaksanakan dan aktor sudah berhasil memasuki sistem.
Basic Path	1. Sistem menampilkan antarmuka yang berisi fungsi reorganisasi keranjang belanja. 2. Aktor memilih batas waktu keranjang belanja yang akan dihapus. 3. Aktor meminta sistem untuk menghapus keranjang belanja yang sudah melebihi batas waktu. 4. Sistem memeriksa keranjang belanja yang sudah melebihi batas waktu. 5. Sistem menghapus keranjang belanja yang bersangkutan.
Alternative Path	-
Postconditions	Keranjang belanja yang telah melebihi batas waktu berhasil dihapus.
Exception Paths	1. Tidak ada keranjang belanja yang melebihi batas waktu (setelah Tabel 3.7 Basic Path 4) a. Sistem menampilkan peringatan bahwa query result kosong.

	b. Kembali ke Tabel 3.7 Basic Path 2.
Extends	-
Includes	-Use Case: Login

3.1.1.8 Spesifikasi Use Case: Pengelolaan Order

Tabel 3.8 Spesifikasi Use Case: Pengelolaan Order

Use Case ID	UC- MERISKA -08
Use Case Name	Pengelolaan Order
Use Case Type	-
Priority	-
Actors	Administrator
Description	Use case ini digunakan untuk mengorganisasi data order buku yang tersimpan di database. Pengorganisasian disini yaitu mengatur status order.
Preconditions	Use case: Login sudah dilaksanakan dan aktor sudah berhasil memasuki sistem.
Basic Path	1. Sistem menampilkan antarmuka yang berisi data-data order. 2. Aktor memilih data order yang ingin diubah status-nya. 3. Aktor mengubah status order. 4. Aktor meminta sistem untuk menyimpan status order yang baru. 5. Sistem menyimpan status order yang baru.
Alternative Path	-
Postconditions	Status order berubah
Exception Paths	-
Extends	-
Includes	-Use Case: Login

3.1.1.9 Spesifikasi Use Case: Pengelolaan Klaster

Tabel 3.9 Spesifikasi Use Case: Pengelolaan Klaster

Use Case ID	UC- MERISKA -09
Use Case Name	Pengelolaan Klaster
Use Case Type	-
Priority	-
Actors	Administrator

Description	Use case ini digunakan untuk mengedit label klaster serta melihat anggota-anggota dalam klaster dan karakteristik klaster. Untuk proses klasterisasinya sendiri dilakukan oleh use case klasterisasi
Preconditions	Use Case: Login dan use case: klasterisasi sudah dilaksanakan dan aktor sudah berhasil memasuki sistem.
Basic Path	1. Sistem memberikan antarmuka untuk mengubah label dari klaster-klaster yang telah terbentuk. 2. Aktor mengedit label dari klaster. 3. Aktor meminta sistem untuk mengubah label dari klaster dengan memicu aksi ubah. 4. Sistem memeriksa data yang diubah oleh aktor. 5. Sistem menyimpan-nya di basis data.
Alternative Path	**A-1.Pengelolaan Informasi** 1. Sistem menampilkan antarmuka yang berisi fungsi untuk mengirim informasi kepada customer berdasarkan klaster. 2. Aktor menginputkan informasi ke dalam form yang disediakan sistem dan memilih klaster yang akan diberikan informasi tersebut. 3. Aktor meminta sistem untuk mengirim informasi tersebut ke customer. 4. Sistem memeriksa informasi. 5. Sistem mengirim informasi ke customer. **A-2.Pengelolaam Diskon** 1. Sistem menampilkan antarmuka yang berisi fungsi untuk memanagement diskon. 2. Aktor memilih buku yang akan diberi diskon. 3. Aktor menginputkan diskon dan klaster-klaster yang mendapat diskon dari buku tersebut. 4. Aktor meminta sistem untuk menyimpan data tersebut ke basis data. 5. Sistem memeriksa data yang diinputkan. 6. Sistem menyimpan data tersebut dalam basis data. **A-3.Pengiriman Informasi Diskon** 1. Sistem menampilkan antarmuka yang berisi list buku yang mempunyai diskon. 2. Aktor memilih salah satu data dalam list.

	3. Aktor meminta sistem untuk mengirim data tersebut ke customer. 4. Sistem mengirim data tersebut ke customer.
Postconditions	Pengelolaan klaster berhasil dilakukan
Exception Paths	1. Label klaster sama (setelah Tabel 3.9 Basic Path 3) a. Sistem menampilkan peringatan bahwa label klaster sama. b. Kembali ke Tabel 3.9 Basic Path 2. 2. Informasi kosong (setelah Tabel 3.9 Alternative Path 1.4) a. Sistem menampilkan peringatan bahwa informasi kosong. b. Kembali ke Tabel 3.9 Alternative Path 1.2. 3. Diskon bernilai 0 (setelah Tabel 3.9 Alternative Path 2.5) a. Sistem menampilkan peringatan bahwa diskon harus lebih besar dari 0. b. Kembali ke Tabel 3.9 Alternative Path 2.3.
Extends	-
Includes	Use Case: Login

3.1.1.10 Spesifikasi Use Case: Notifikasi Pembayaran

Tabel 3.10 Spesifikasi Use Case: Notifikasi Pembayaran

Use Case ID	UC- MERISKA -10
Use Case Name	Notifikasi Pembayaran
Use Case Type	-
Priority	-
Actors	Customer
Description	Use case ini dilakukan oleh customer untuk memberitahukan kepada sistem bahwa proses pembayaran telah dilakukan kemudian sistem akan menyampaikan informasi tersebut kepada *administrator* dalam bentuk email notifikasi pembayaran.
Preconditions	Use case: Login telah dilakukan dan aktor telah masuk ke dalam sistem.
Basic Path	1. Sistem menampilkan antarmuka untuk melakukan notifikasi pembayaran. 2. Aktor memilih order.

	3. Aktor meminta sistem untuk menampilkan detail order yang dipilih. 4. Sistem mengambil data detail order dari basis data dan menampilkannya ke aktor. 5. Aktor meminta sistem untuk mengirim konfirmasi pembayaran ke administrator. 6. Sistem mengirim email konfirmasi pembayaran ke administrator.
Alternative Path	**A-1.Cancel Order** 1. Sistem menampilkan antarmuka untuk melakukan notifikasi pembayaran. 2. Aktor memilih order. 3. Aktor meminta sistem untuk menampilkan detail order yang dipilih. 4. Sistem mengambil data detail order dari basis data dan menampilkannya ke aktor. 5. Aktor meminta sistem untuk membatalkan order. 6. Sistem mengubah status order menjadi cancel order.
Postconditions	Notifikasi pembayaran telah dilakukan dan informasi mengenai pembayaran telah disampaikan kepada administrator.
Exception Paths	-
Extends	-
Includes	-Use Case: Login

3.1.1.11 Spesifikasi Use Case: Tampil Informasi

Tabel 3.11 Spesifikasi Use Case: Tampil Informasi

Use Case ID	UC- MERISKA -11
Use Case Name	Tampil Informasi
Use Case Type	-
Priority	-
Actors	Customer
Description	Use case ini digunakan oleh *customer* untuk melihat informasi-infromasi yang disediakan oleh sistem seperti informasi katalog buku dan informasi buku detail.
Preconditions	-

Basic Path	1. Sistem menampilkan antarmuka yang berisi informasi-informasi mengenai katalog buku dan buku detail. 2. Aktor memilih infomasi yang ingin dilihat. 3. Sistem menampilkan detail informasi tersebut.
Alternative Path	-
Postconditions	Informasi dapat diakses oleh aktor.
Exception Paths	-
Extends	-
Includes	-

3.1.1.12 Spesifikasi Use Case: Pengelolaan Email

Tabel 3.12 Spesifikasi Use Case: Pengelolaan Email

Use Case ID	UC- MERISKA -12
Use Case Name	Pengelolaan Email
Use Case Type	-
Priority	-
Actors	Administrator
Description	Use case ini digunakan oleh *administrator* untuk mengirim email-email notifikasi ke customer berdasarkan proses order yang terjadi.
Preconditions	- Use case: Login telah dilakukan.
Basic Path	1. Sistem menampilkan antarmuka yang berisi detail order dan email notifikasi yang belum dikirim ke customer. 2. Aktor memilih order yang akan dikirim email notifikasi. 3. Aktor meminta sistem untuk mengirimkan email notifikasi ke customer dengan memicu aksi kirim. 4. Sistem mengirimkan email notifikasi ke customer.
Alternative Path	-
Postconditions	Email notifikasi telah dikirimkan ke customer.
Exception Paths	-
Extends	-
Includes	- Use case: Login

3.1.1.13 Spesifikasi Use Case: Pengelolaan Stok

Tabel 3.13 Spesifikasi Use Case: Pengelolaan Stok

Use Case ID	UC- MERISKA -13

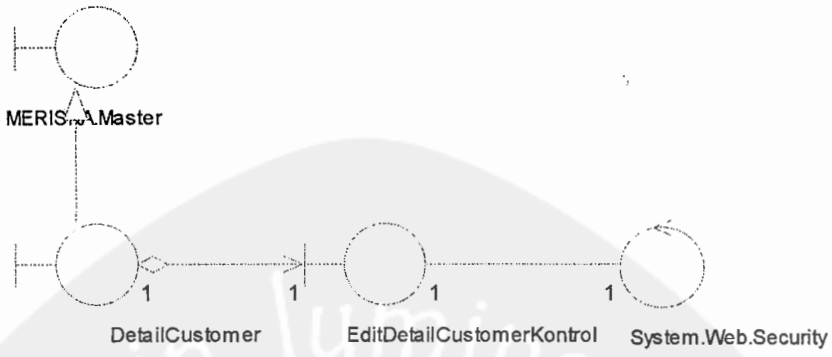
Use Case Name	Pengelolaan Stok
Use Case Type	-
Priority	-
Actors	Administrator
Description	Use case ini digunakan oleh *administrator* untuk mengatur stok buku dalam katalog.
Preconditions	- Use case: Login telah dilakukan.
Basic Path	1. Sistem menampilkan antarmuka yang berisi edit stok buku. 2. Aktor memasukan judul buku yang ingin diubah stoknya. 3. Aktor mengubah stoknya. 4. Aktor meminta sistem untuk menyimpan perubahannya dengan memicu aksi ubah. 5. Sistem menyimpan data perubahan ke basisdata.
Alternative Path	-
Postconditions	Stok buku berubah.
Exception Paths	1. Stok buku yang dimasukan lebih kecil dari 1 (setelah Tabel 3.13 Basic Path 4) a. Sistem menampilkan peringatan bahwa stok buku yang diisikan harus lebih besar dari 0. b. Kembali ke Tabel 3.13 Basic Path 3.
Extends	-
Includes	- Use case: Login

3.1.2 Static Structure Diagram

3.1.2.1 Analysis Class Diagram: Use Case: Login

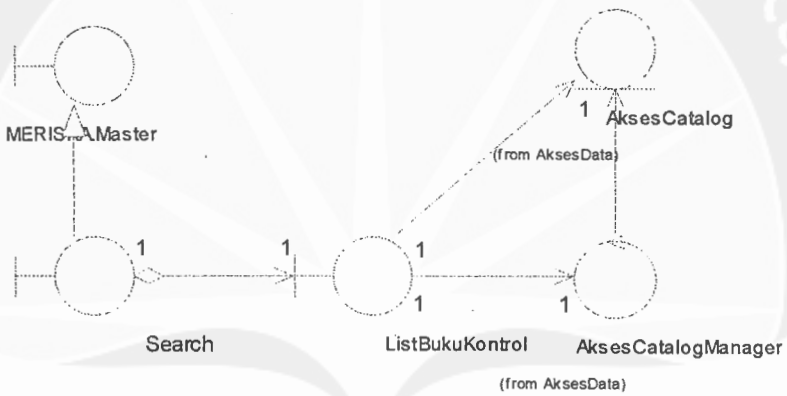
Gambar 3.1 Analysis Class Diagram : Use Case: Login

3.1.2.2 Analysis Class Diagram: Use Case: Pengubahan Profil Customer



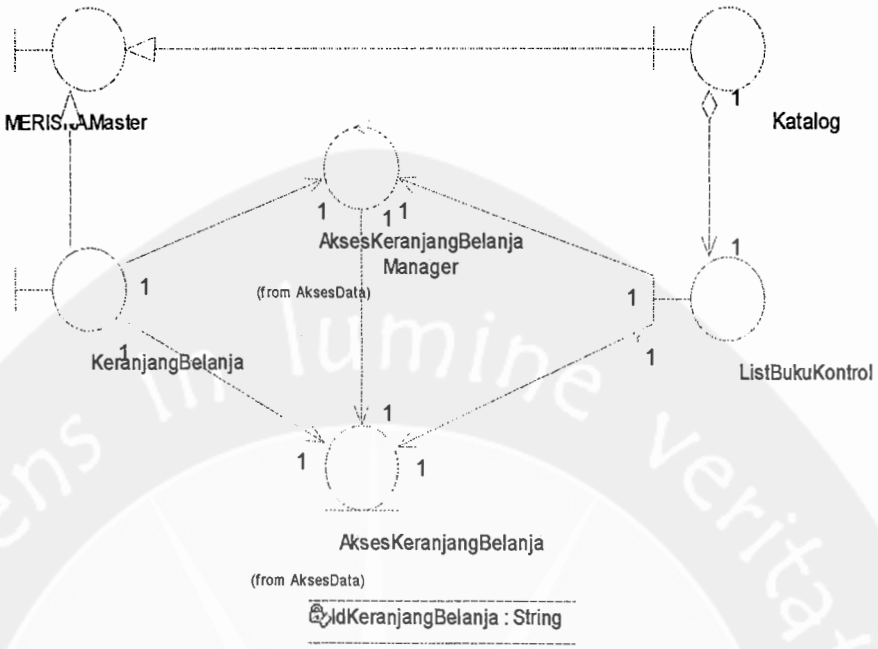
Gambar 3.2 Analysis Class Diagram: Use Case: Pengubahan Profil Customer

3.1.2.3 Analysis Class Diagram: Use Case: Pencarian Catalog



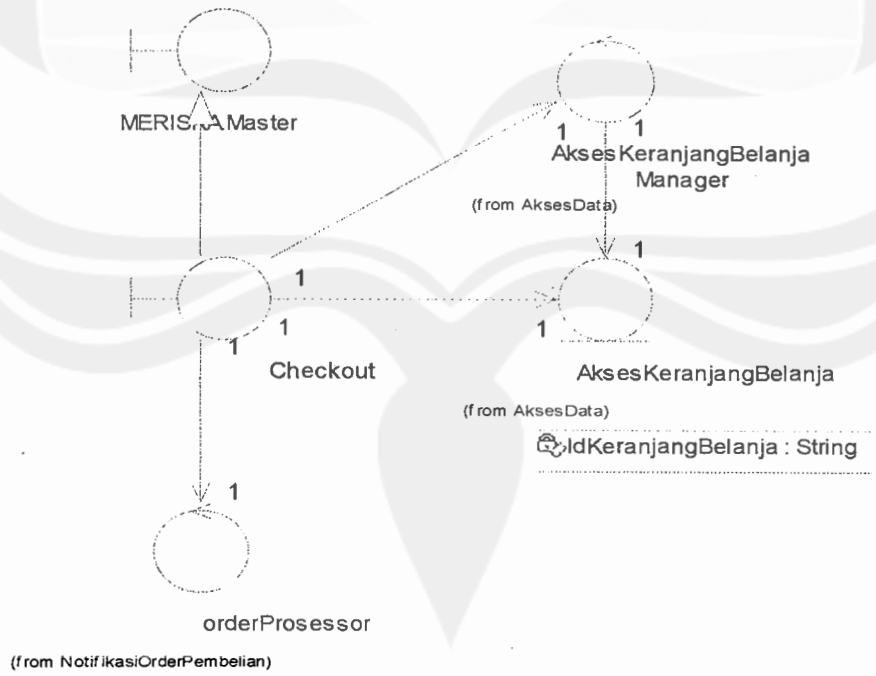
Gambar 3.3 Analysis Class Diagram: Use Case: Pencarian Catalog

3.1.2.4 Analysis Class Diagram: Use Case: Pengisian Keranjang Belanja



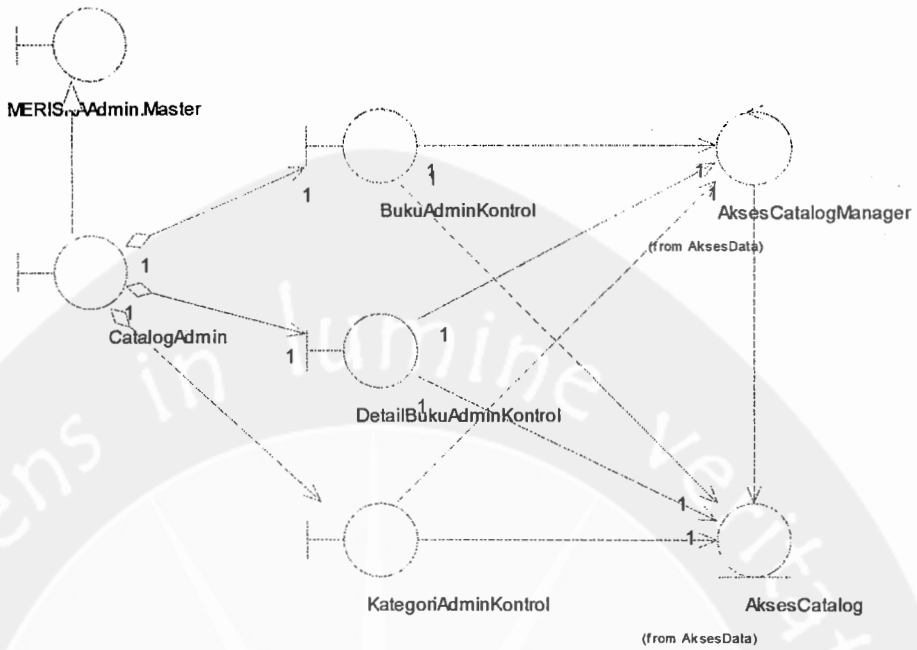
Gambar 3.4 Analysis Class Diagram: Use Case: Pengisian Keranjang Belanja

3.1.2.5 Analysis Class Diagram: Use Case: Order



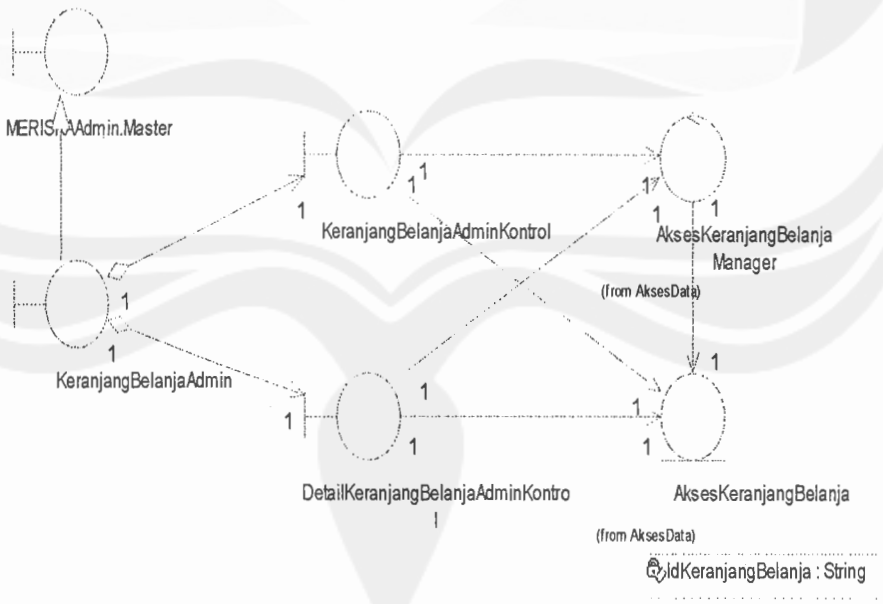
Gambar 3.5 Analysis Class Diagram: Use Case: Order

3.1.2.6 Analysis Class Diagram: Use Case: Pengelolaan Catalog



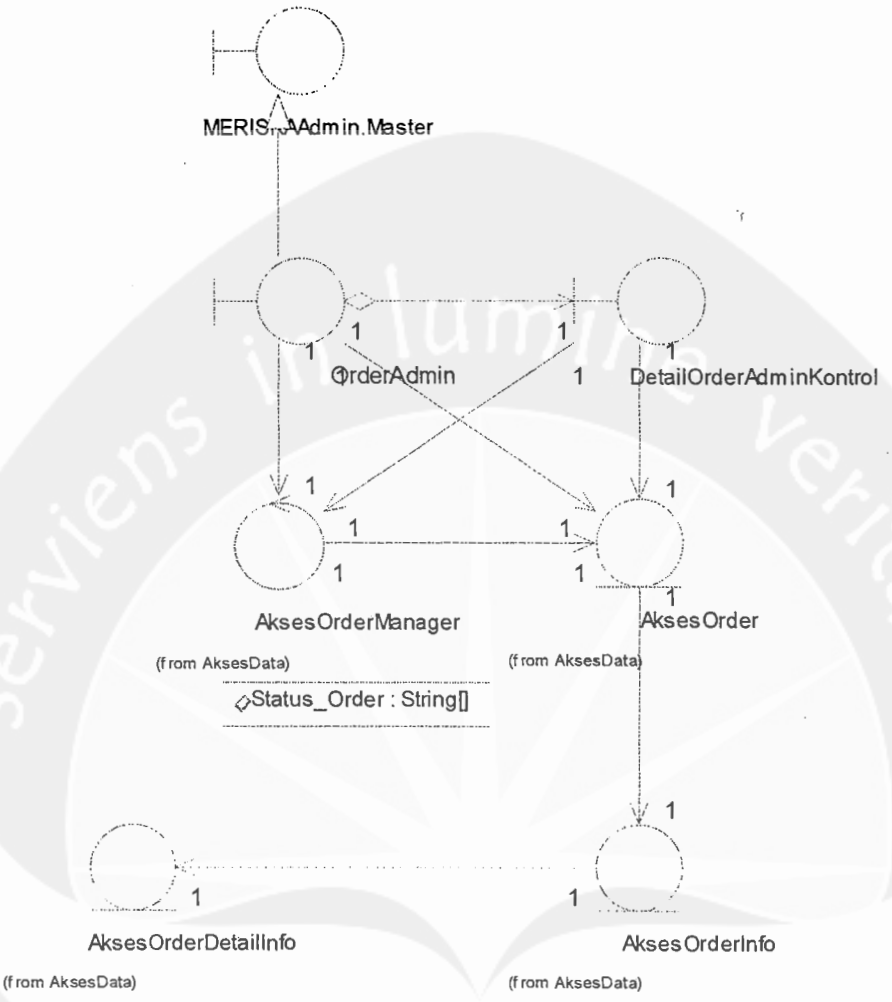
Gambar 3.6 Analysis Class Diagram: Use Case: Pengelolaan Catalog

3.1.2.7 Analysis Class Diagram: Use Case: Pengelolaan Keranjang Belanja



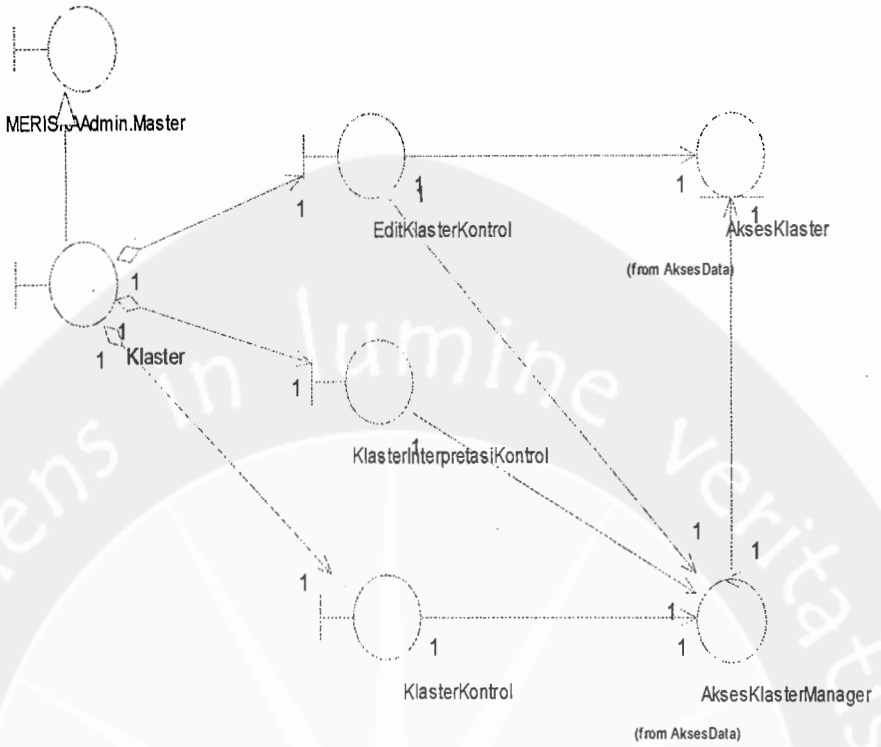
Gambar 3.7 Analysis Class Diagram: Use Case: Pengelolaan Keranjang Belanja

3.1.2.8 Analysis Class Diagram: Use Case: Pengelolaan Order



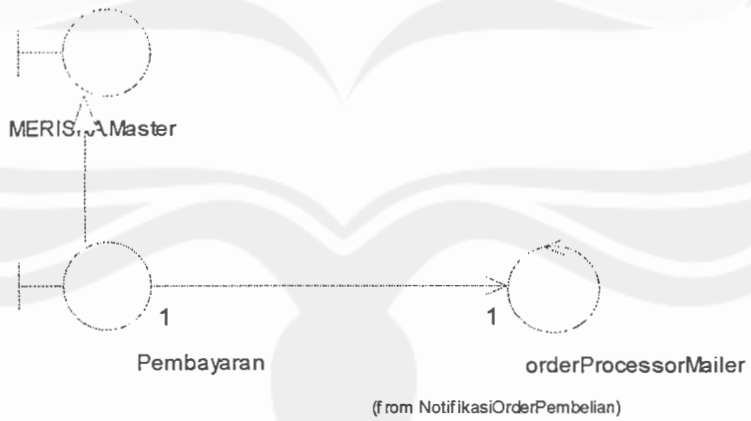
Gambar 3.8 Analysis Class Diagram: Use Case: Pengelolaan Order

3.1.2.9 Analysis Class Diagram: Use Case: Pengelolaan Klaster



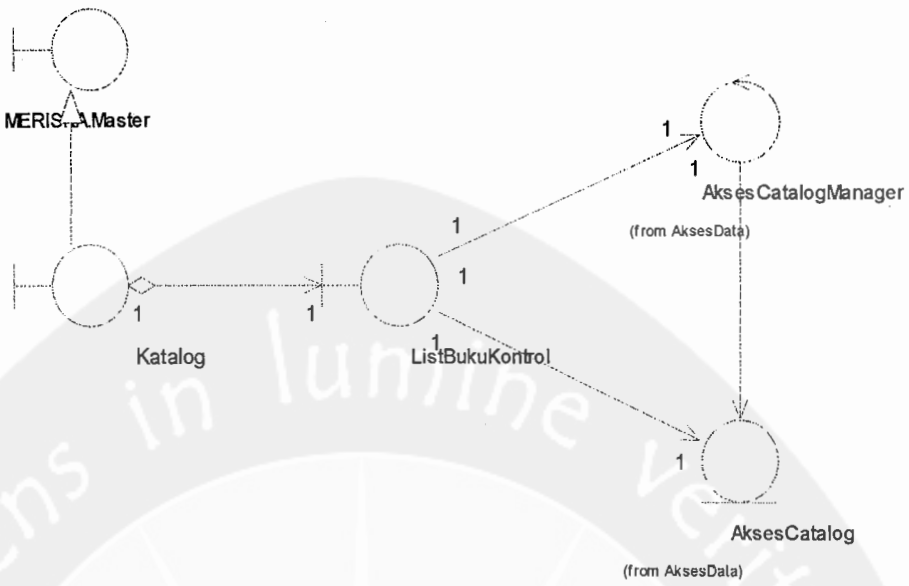
Gambar 3.9 Analysis Class Diagram: Use Case: Pengelolaan Klaster

3.1.2.10 Analysis Class Diagram: Use Case: Notifikasi Pembayaran



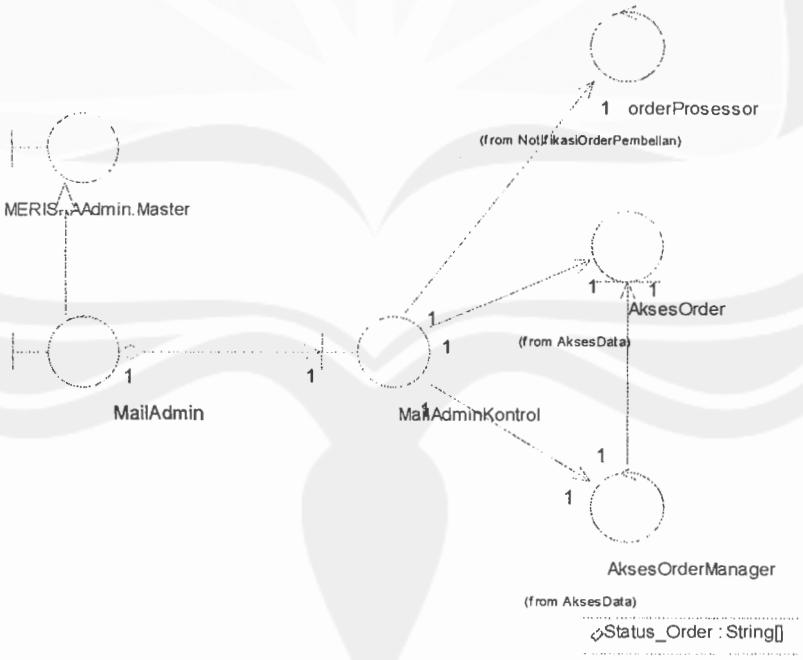
Gambar 3.10 Analysis Class Diagram: Use Case: Notifikasi Pembayaran

3.1.2.11 Analysis Class Diagram: Use Case: Tampil Informasi



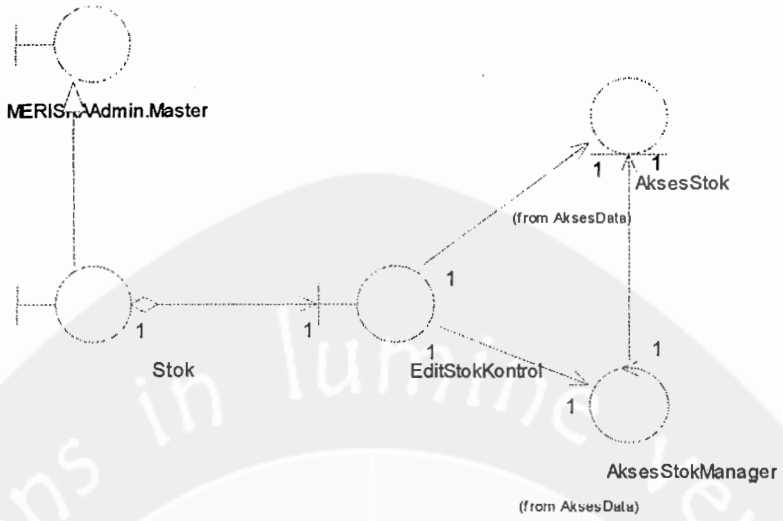
Gambar 3.11 Analysis Class Diagram: Use Case: Tampil Informasi

3.1.2.12 Analysis Class Diagram: Use Case: Pengelolaan Email



Gambar 3.12 Analysis Class Diagram: Use Case: Pengelolann Email

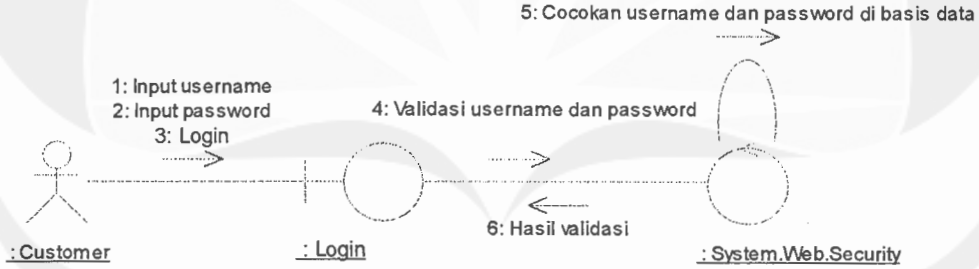
3.1.2.13 Analysis Class Diagram: Use Case: Pengelolaan Stok



Gambar 3.13 Analysis Class Diagram: Use Case: Pengelolaan Stok

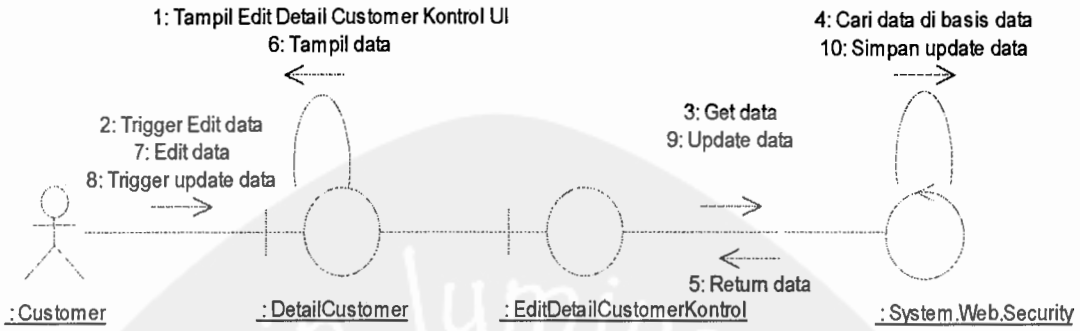
3.1.3 Interaction Diagram

3.1.3.1 Analysis Collaboration Diagram: Use Case Login



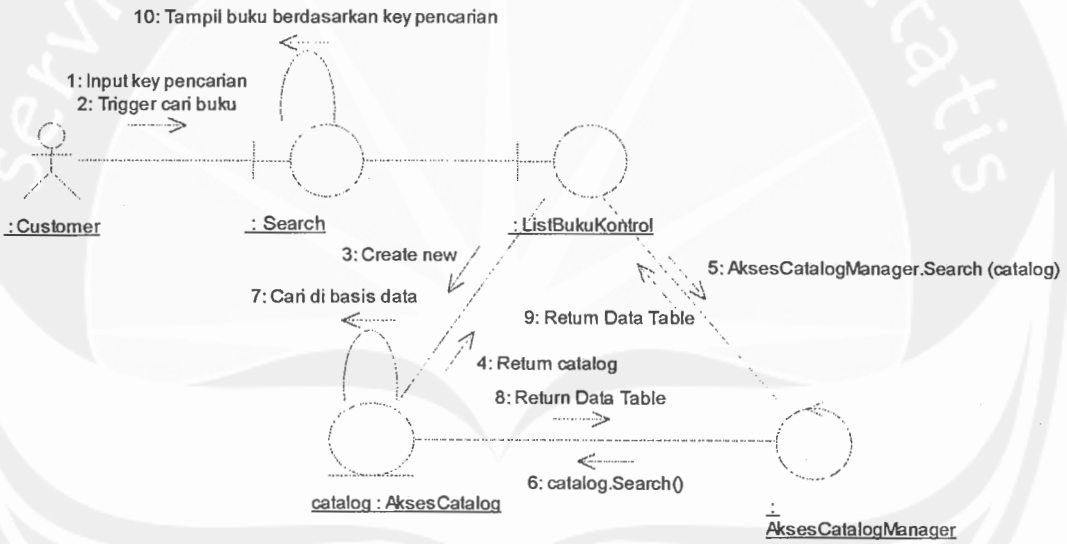
Gambar 3.14 Analysis Collaboration Diagram: Use Case Login

3.1.3.2 Analysis Collaboration Diagram: Use Case Pengubahan Profil Customer



Gambar 3.15 Analysis Collaboration Diagram: Use Case Pengubahan Profil Customer

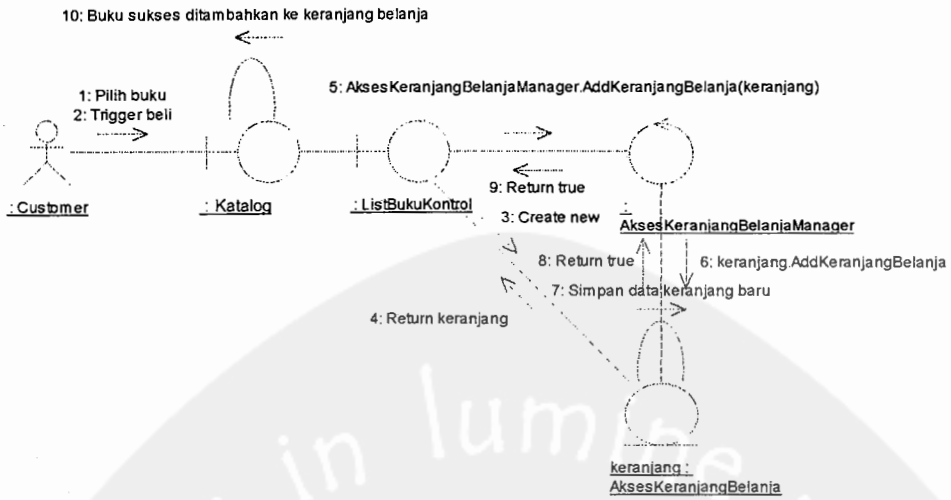
3.1.3.3 Analysis Collaboration Diagram: Use Case Pencarian Catalog



Gambar 3.16 Analysis Collaboration Diagram: Use Case Pencarian Catalog

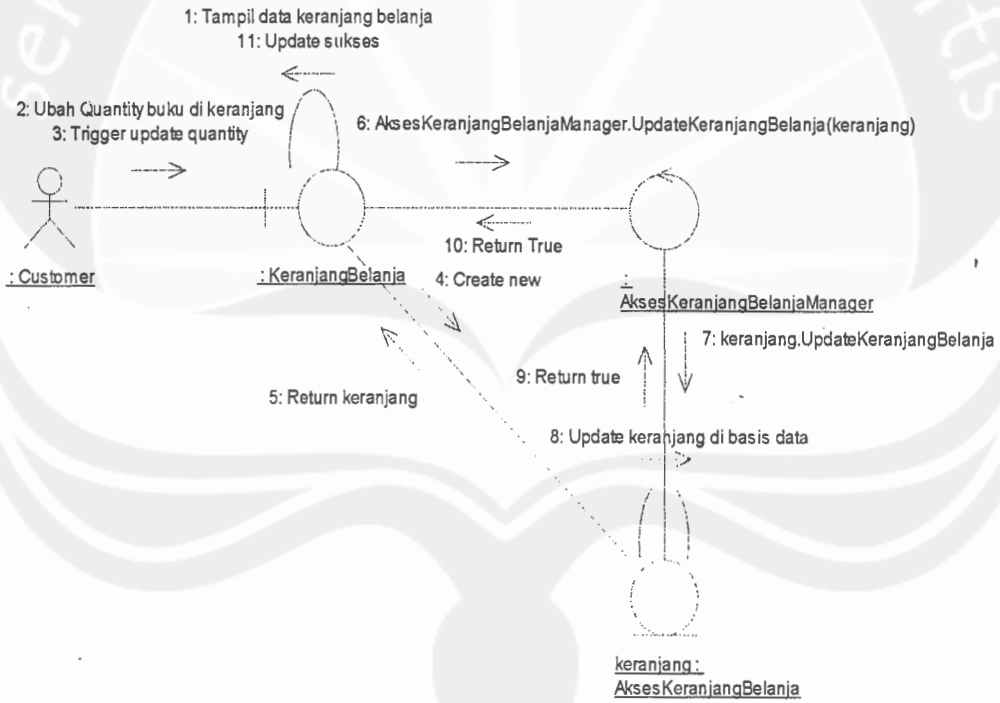
3.1.3.4 Analysis Collaboration Diagram: Use Case Pegisian Keranjang Belanja

3.1.3.4.1 Tambah data buku ke keranjang belanja



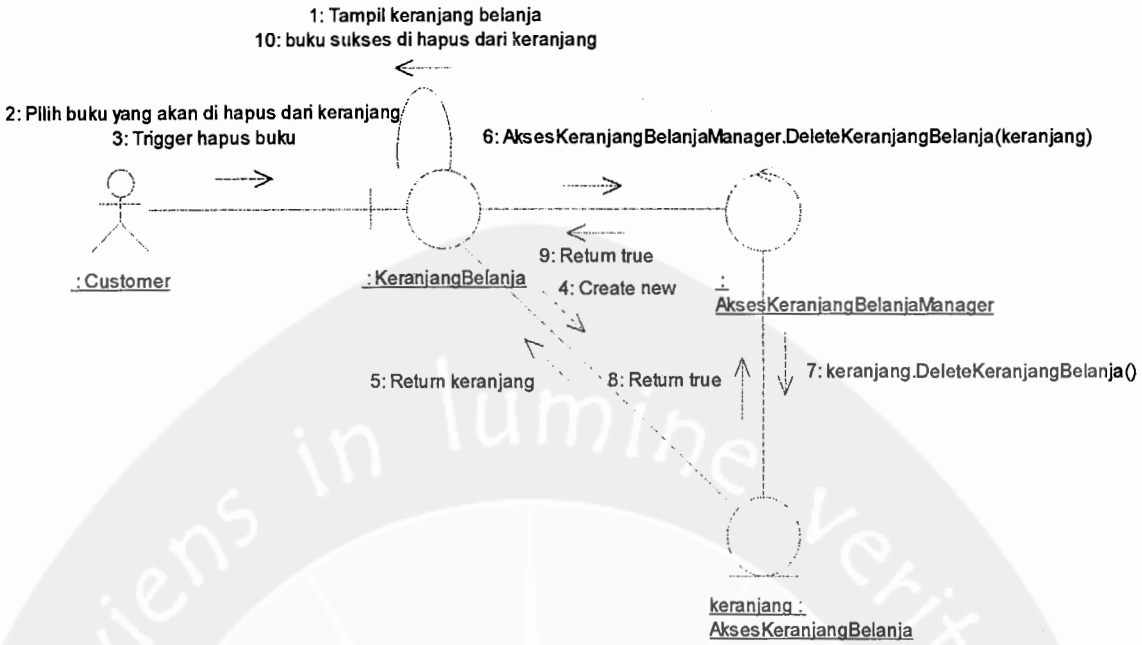
Gambar 3.17 Analysis Collaboration Diagram : Use Case Pengisian Keranjang Belanja – Tambah data buku ke keranjang belanja

3.1.3.4.2 Ubah Jumlah Buku di Keranjang Belanja



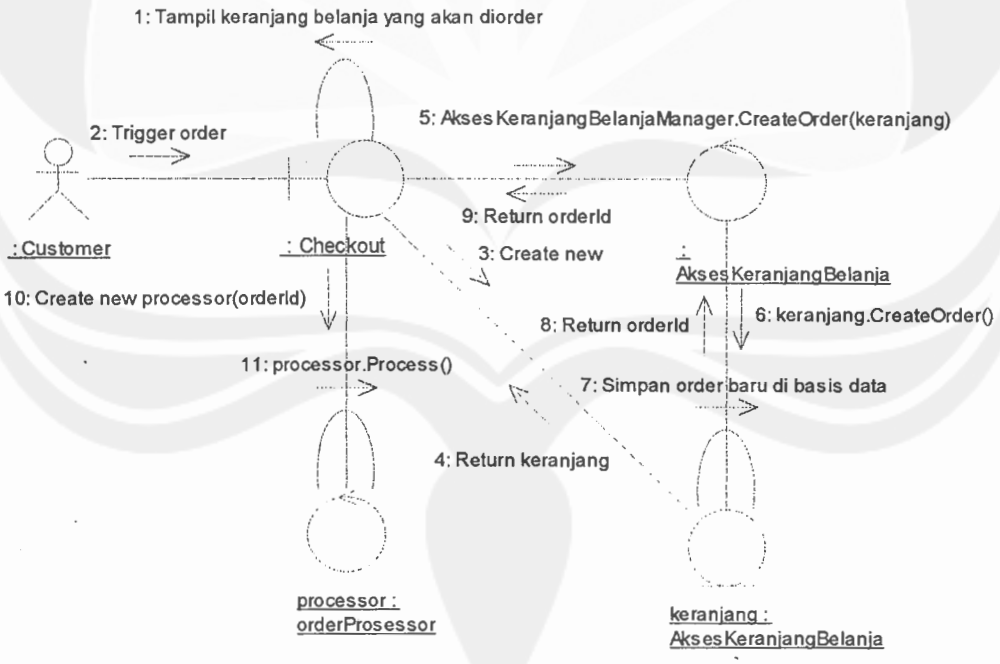
Gambar 3.18 Analysis Collaboration Diagram: Use Case Pengisian Keranjang Belanja – Ubah Jumlah Buku di Keranjang Belanja

3.1.3.4.3 Hapus Buku dari Keranjang Belanja



Gambar 3.19 Analysis Collaboration Diagram: Use Case Pengisian Keranjang Belanja – Hapus Buku dari Keranjang Belanja

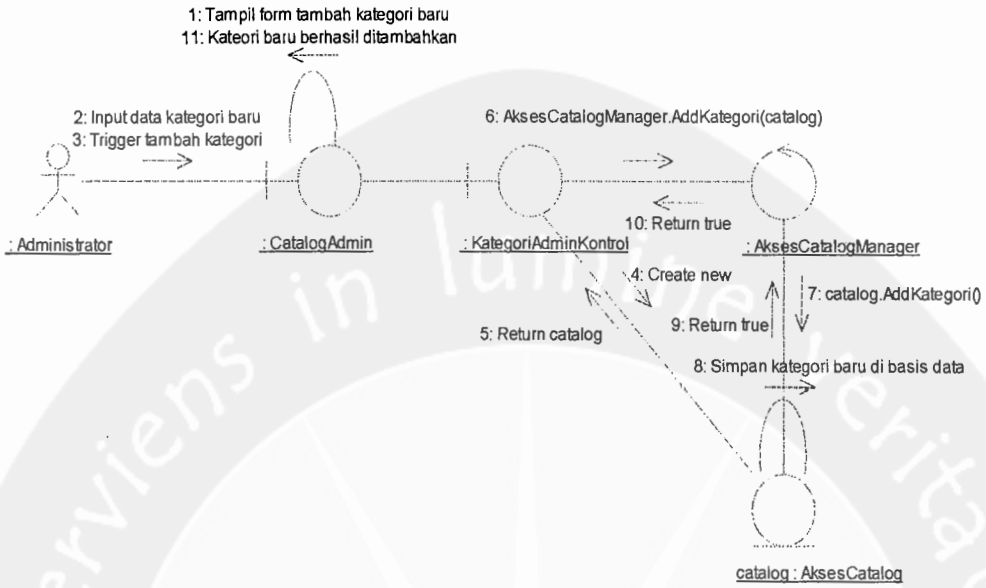
3.1.3.5 Analysis Collaboration Diagram: Use Case Order



Gambar 3.20 Analysis Collaboration Diagram: Use Case Order

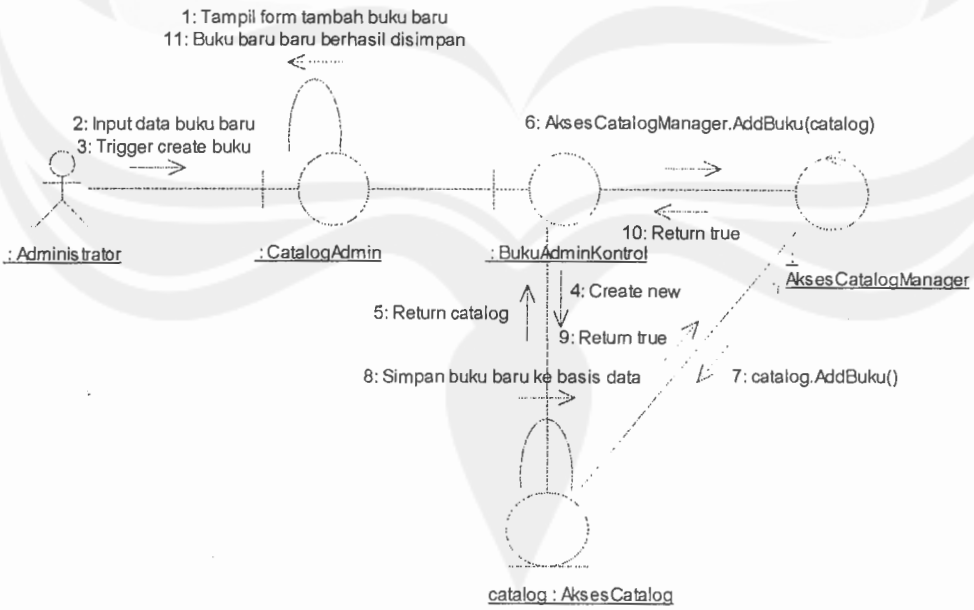
3.1.3.6 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog

3.1.3.6.1 Tambah Kategori Baru



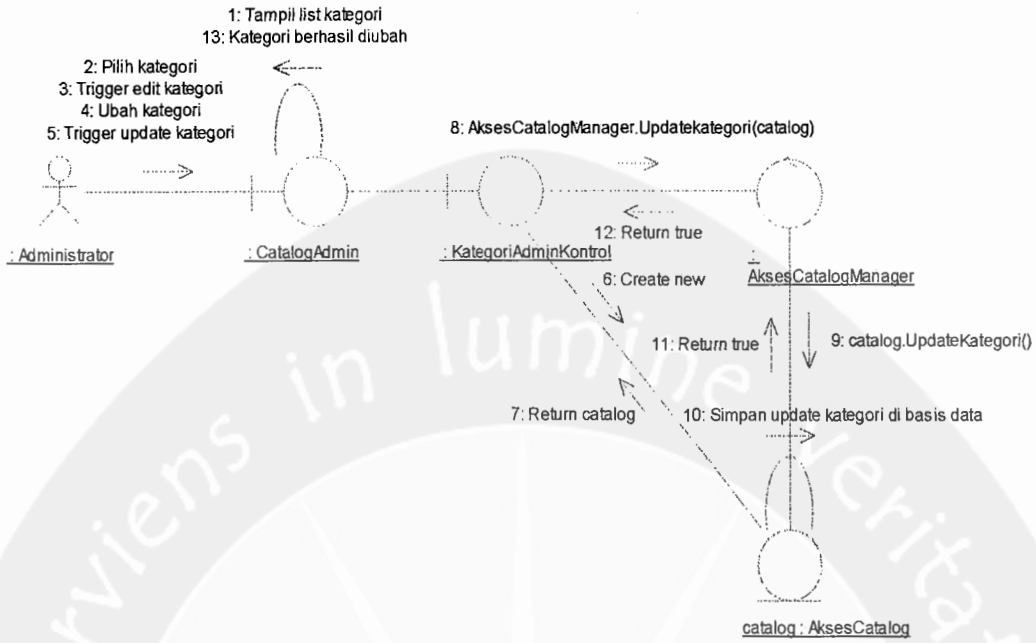
Gambar 3.21 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog – Tambah Kategori Baru

3.1.3.6.2 Tambah Data Buku Baru



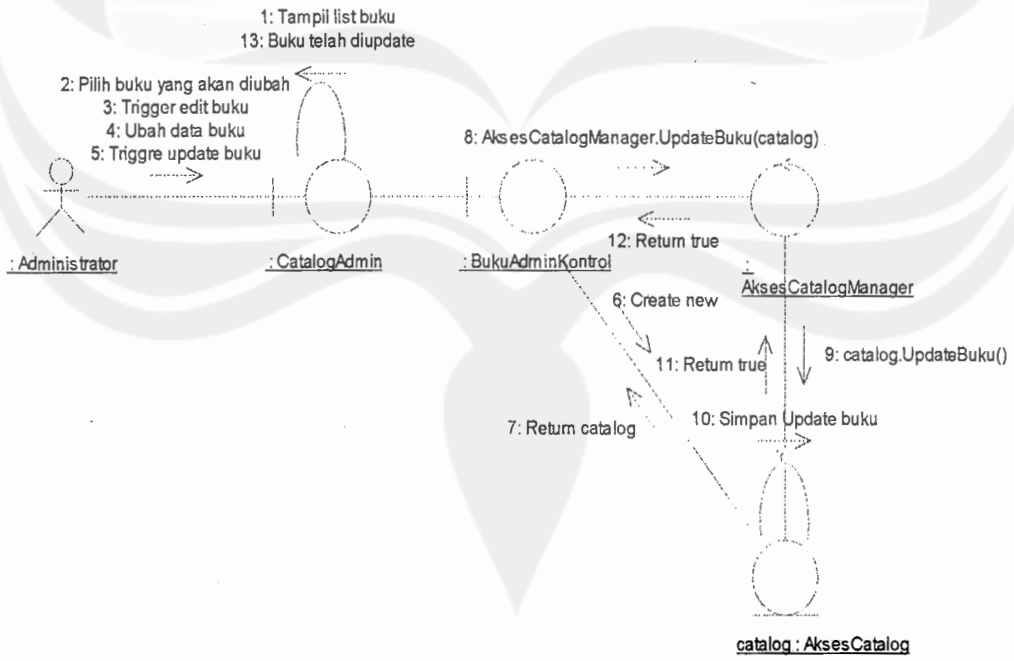
Gambar 3.22 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog – Tambah Data Buku Baru

3.1.3.6.3 Ubah Kategori



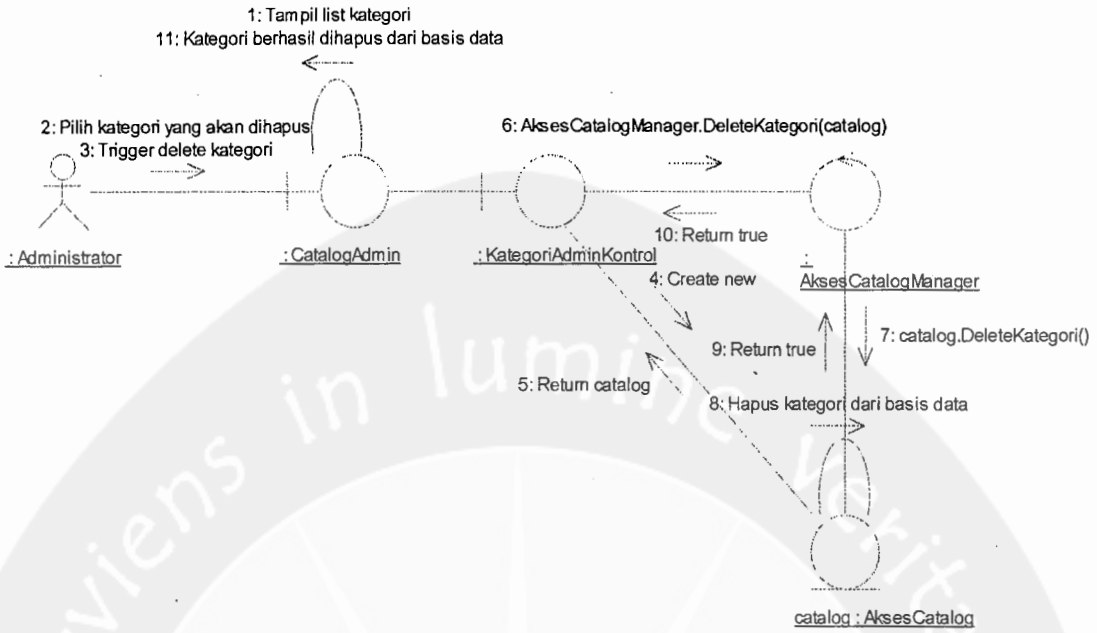
Gambar 3.23 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog – Ubah Kategori

3.1.3.6.4 Ubah Buku



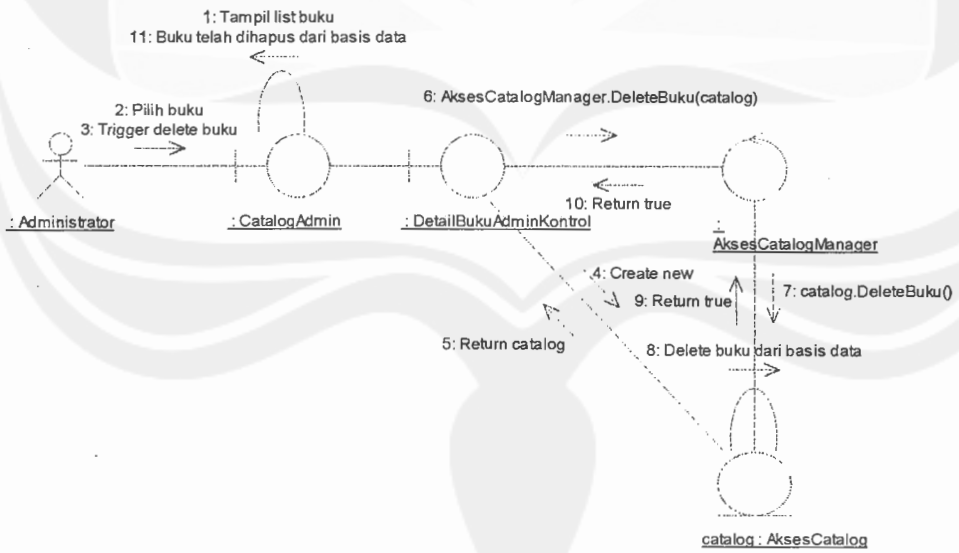
Gambar 3.24 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog– Ubah Buku

3.1.3.6.5 Hapus Kategori



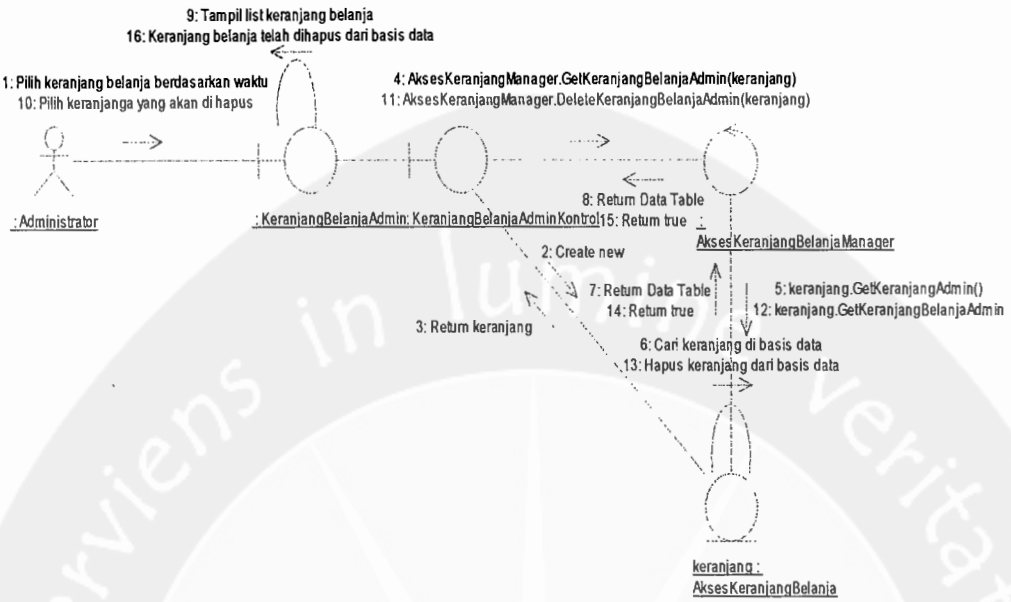
Gambar 3.25 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog – Hapus Kategori

3.1.3.6.6 Hapus Buku



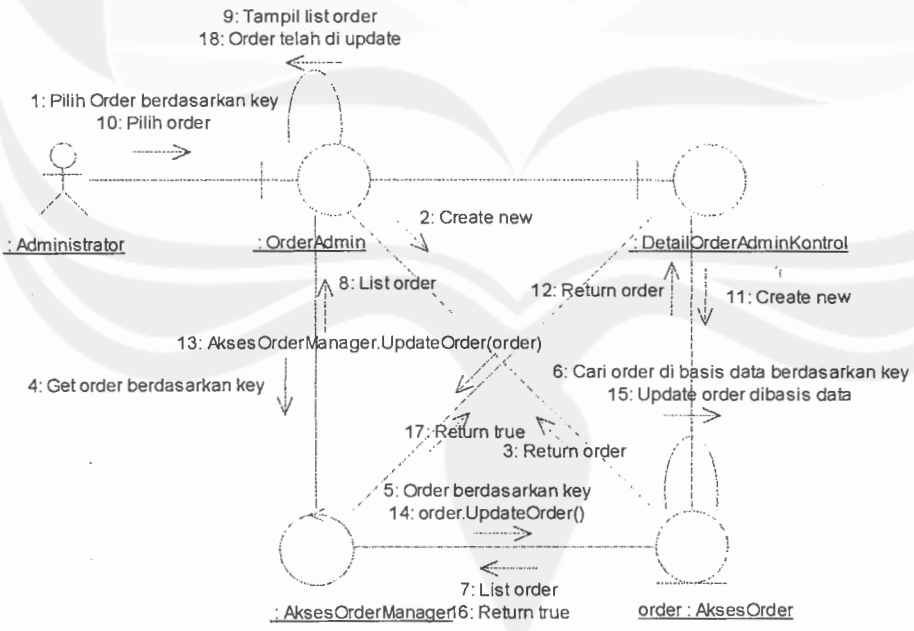
Gambar 3.26 Analysis Collaboration Diagram: Use Case Pengelolaan Catalog – Hapus Buku

3.1.3.7 Analysis Collaboration Diagram: Use Case Pengelolaan Keranjang Belanja



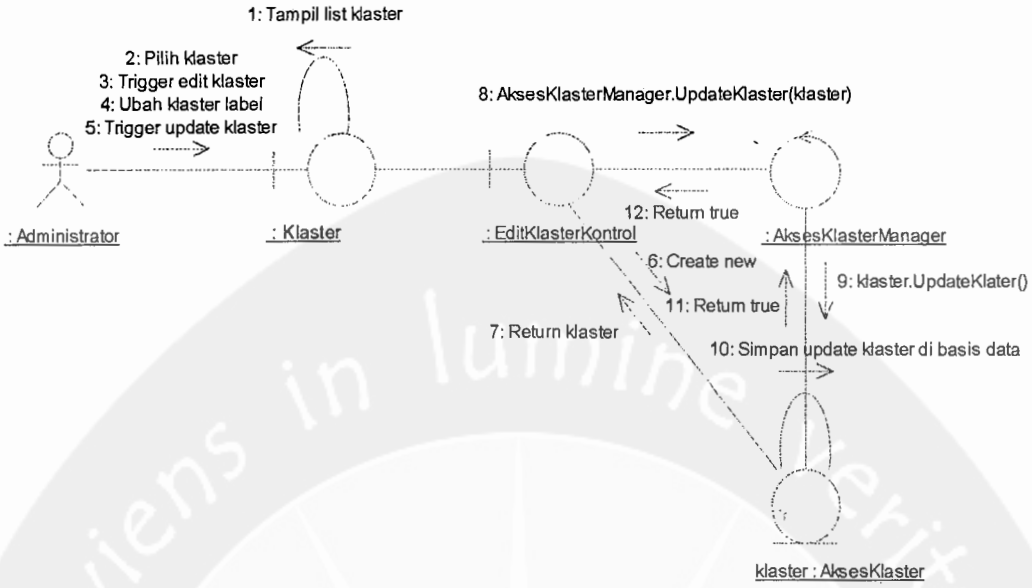
Gambar 3.27 Analysis Collaboration Diagram: Use Case Pengelolaan Keranjang Belanja

3.1.3.8 Analysis Collaboration Diagram: Use Case Pengelolaan Order



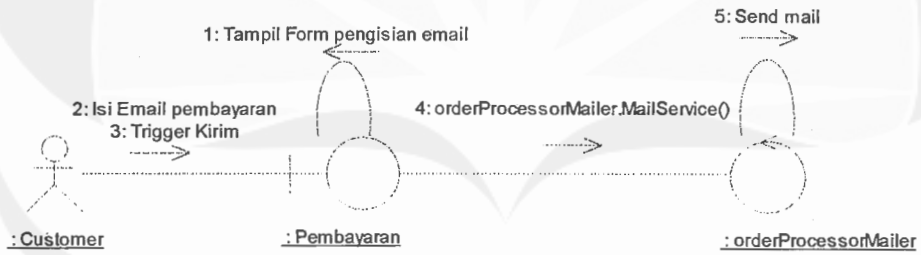
Gambar 3.28 Analysis Collaboration Diagram: Use Case Pengelolaan Order – Notifikasi Email Pembayaran telah diterima

3.1.3.9 Analysis Collaboration Diagram: Use Case Pengelolaan Klaster



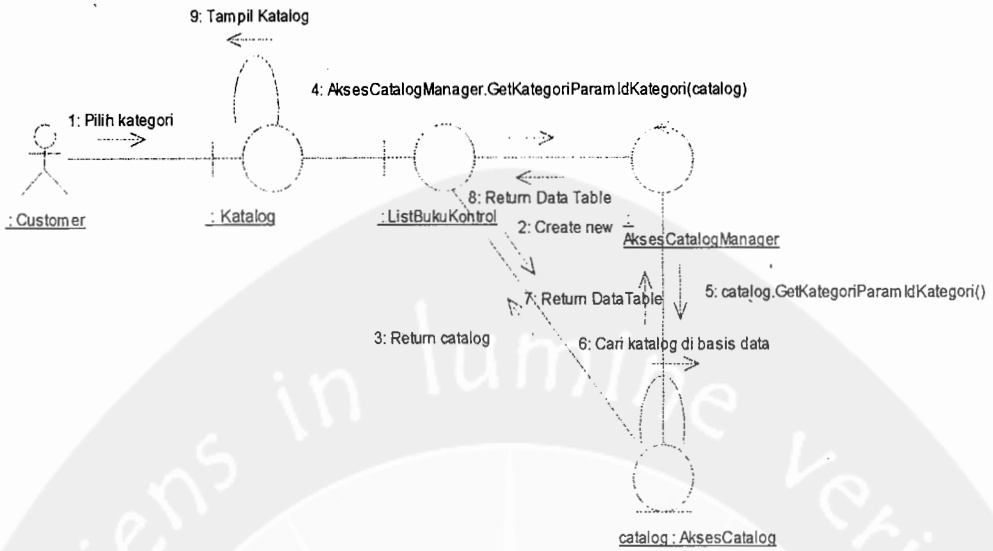
Gambar 3.29 Analysis Collaboration Diagram: Use Case Pengelolaan Klaster – Edit Klaster Label

3.1.3.10 Analysis Collaboration Diagram: Use Case Notifikasi Pembayaran



Gambar 3.30 Analysis Collaboration Diagram: Use Email Pembayaran – Notifikasi Pembayaran

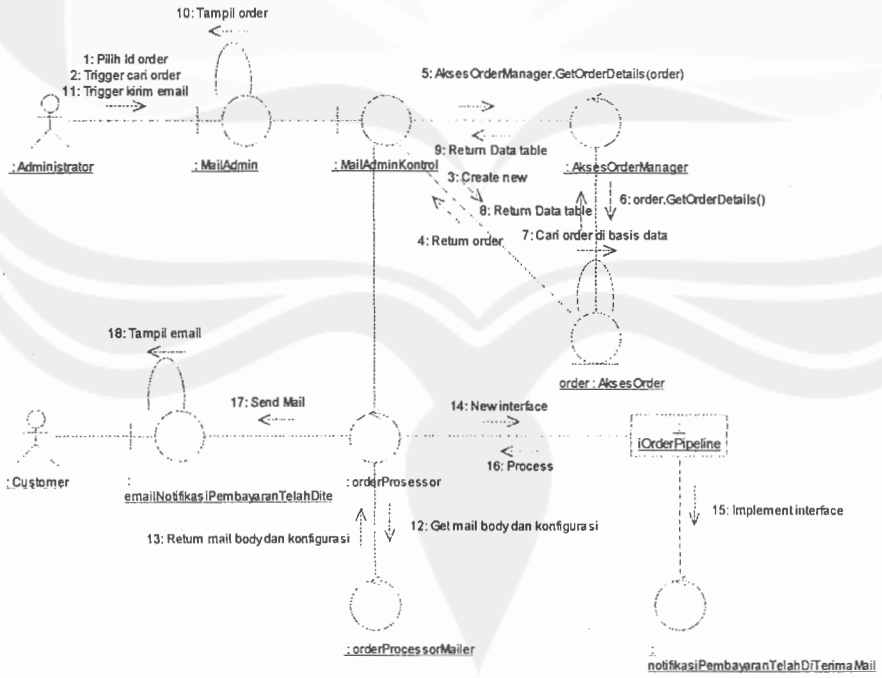
3.1.3.11 Analysis Collaboration Diagram: Use Case Tampil Informasi



Gambar 3.31 Analysis Collaboration Diagram: Use Email Tampil Informasi

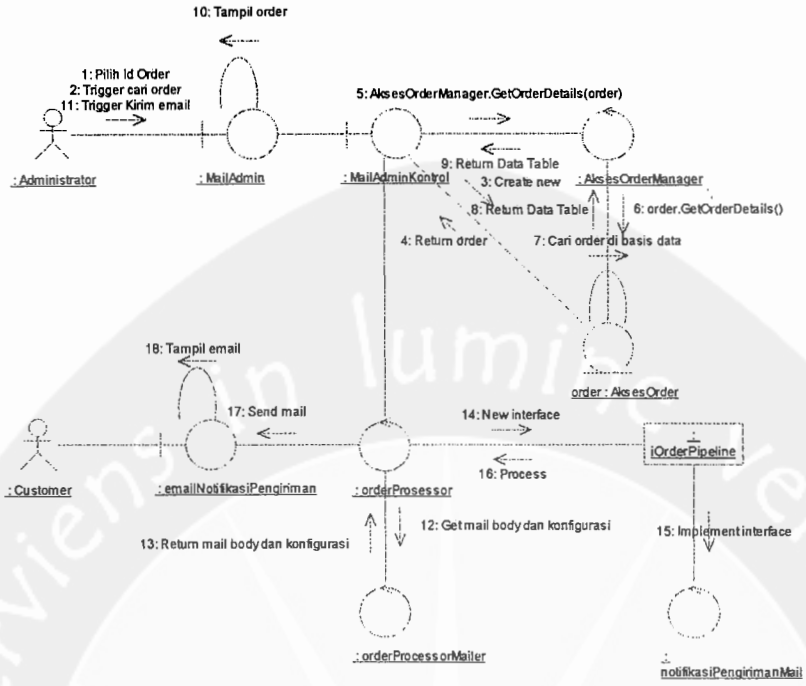
3.1.3.12 Analysis Collaboration Diagram: Use Case Pengelolaan Email

3.1.3.12.1 Email notifikasi pembayaran telah diterima



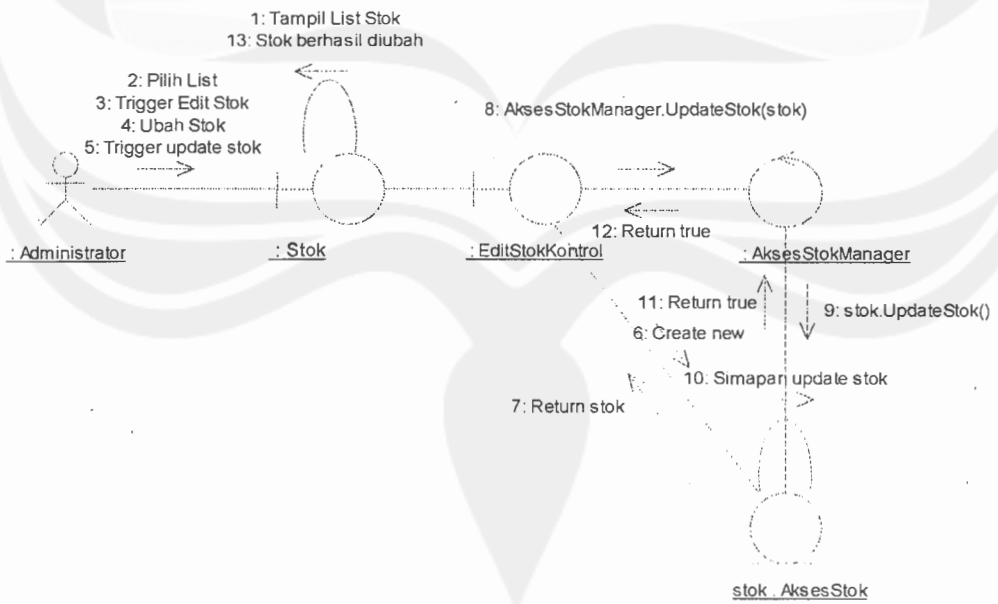
Gambar 3.32 Analysis Collaboration Diagram: Use Case Pengelolaan Email – Email Notifikasi pembayaran telah diterima

3.1.3.12.2 Email notifikasi pengiriman



Gambar 3.33 Analysis Collaboration Diagram: Use Case Pengelolaan Email – Email Notifikasi pengiriman

3.1.3.13 Analysis Collaboration Diagram: Use Case Pengelolaan Stok



Gambar 3.34 Analysis Collaboration Diagram: Use Case Pengelolaan Stok

3.2 Spesifikasi Kebutuhan Non-Fungsionalitas

3.2.1 Kebutuhan Antarmuka Eksternal

Kebutuhan antar muka eksternal pada perangkat lunak MERISKA meliputi kebutuhan antarmuka pemakai, antarmuka perangkat keras, antarmuka perangkat lunak, dan antarmuka komunikasi.

3.2.2 Antarmuka Pemakai

Pengguna berinteraksi dengan antarmuka yang ditampilkan dalam layar komputer dengan format tampilan halaman *web* .Antarmuka ini memungkinkan pengguna baik *customer* maupun *administrator* untuk mengakses fungsionalitas yang diberikan sistem dari lokasi manapun mereka berada selama terkoneksi dengan layanan internet.

3.2.3 Antarmuka Perangkat Keras

Antarmuka perangkat keras yang digunakan dalam pengujian perangkat lunak MERISKA adalah:

1. *PC AMD Atlhon 64*
2. *Keyboard dan mouse*
3. LAN

3.2.4 Antarmuka Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk mengoperasikan perangkat lunak MERISKA adalah:

1. Nama : *SQL Server 2005*
   Sumber : *Microsoft*
   Sebagai basis data yang dibutuhkan dalam mengoperasikan perangkat lunak MERISKA.
2. Nama : *Visual C# .NET 2005, ASP .NET 2.0*
   Sumber : *Microsoft*

Sebagai tool perancangan yang dibutuhkan dalam pengembangkan perangkat lunak MERISKA.

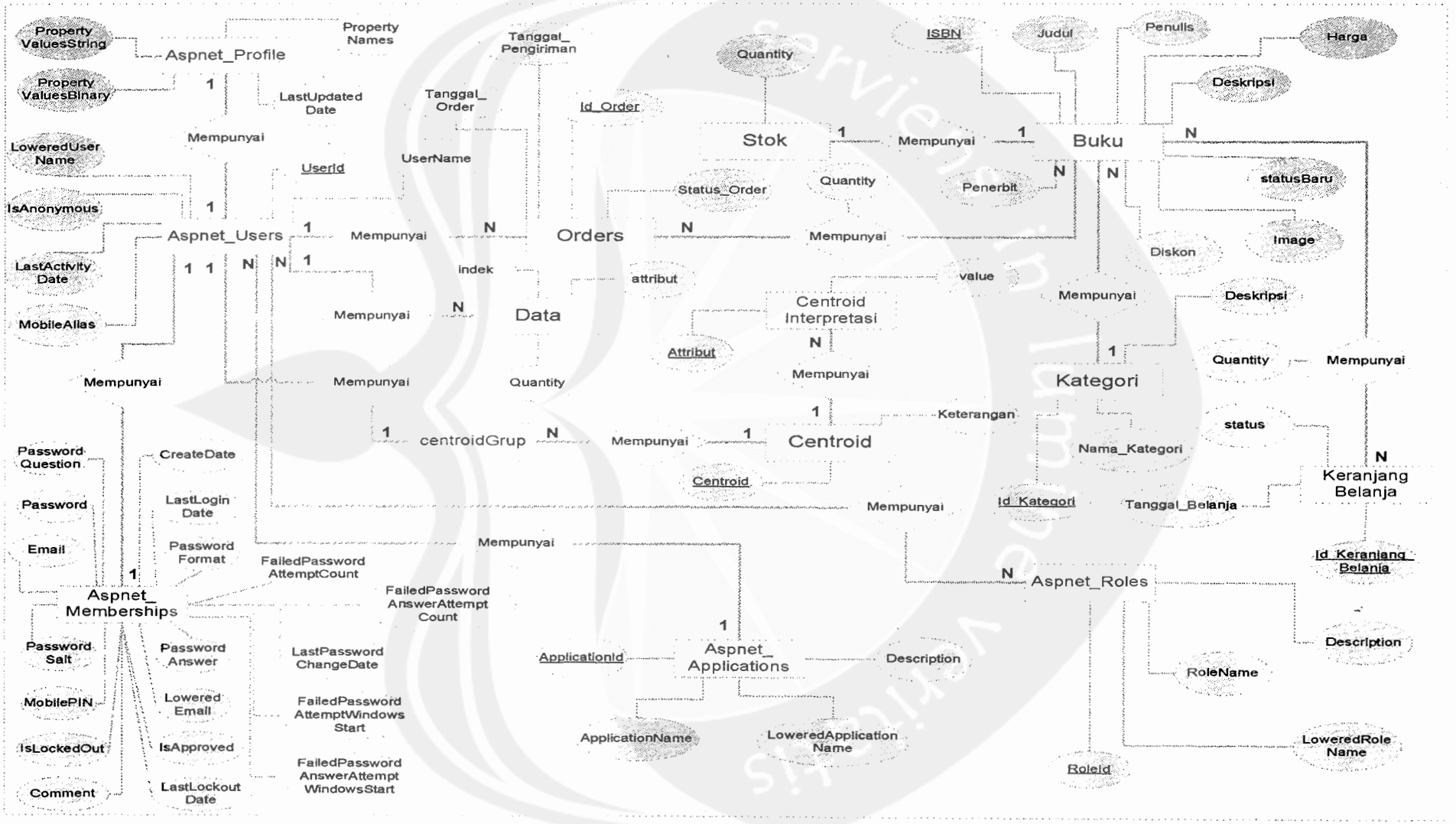
3. Nama : *Internet Explorer 5.5/6.0*
   Sumber : *Microsoft*
   Sebagai browser internet yang dibutuhkan oleh pengguna untuk melakukan transaksi.
4. Nama : *Internet Information Services (IIS)*
   Sumber : *Microsoft*
   Sebagai *Web Server*.
5. Nama : *Windows 2003 Server / XP*
   Sumber : *Microsoft*.
   Sebagai sistem operasi komputer.

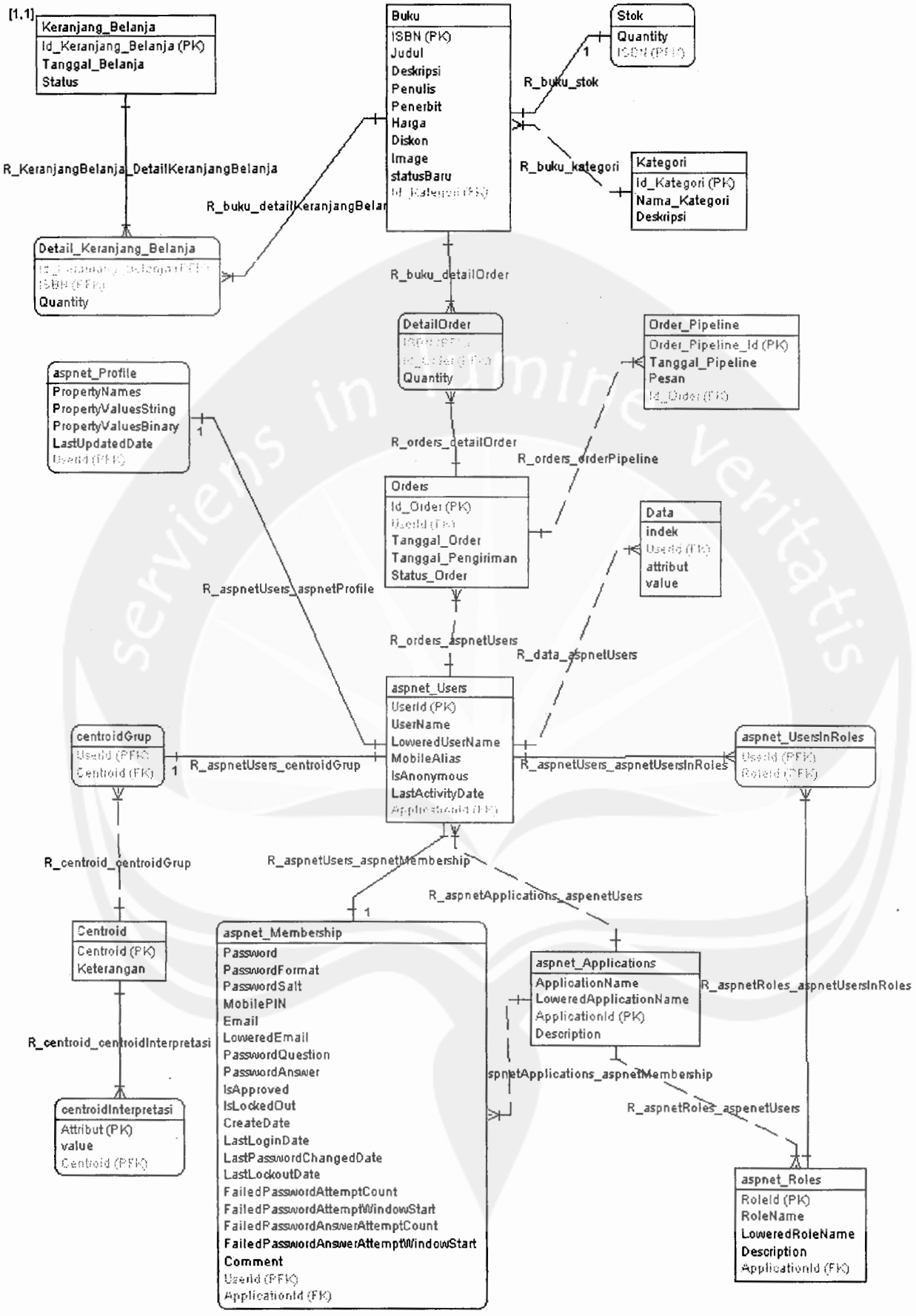
3.2.5 Antarmuka Komunikasi

Antarmuka komunikasi perangkat lunak MERISKA menggunakan protocol TCP/IP karena perangkat lunak ini digunakan untuk mendukung model sistem *client-server* dengan media komunikasi intranet atau internet.

3.3 Spesifikasi Kebutuhan Data

3.3.1 ERD (Entity Relationship Diagram)




[1,1]
Keranjang_Belanja
Id_Keranjang_Belanja (PK)
Tanggal_Belanja
Status
Buku
ISBN (PK)
Judul
Deskripsi
Penulis
Penerbit
Harga
Diskon
Image
statusBaru
Id_Kategori (FK)
Stok
Quantity
ISBN (PFK)
1
R_buku_stok
R_KeranjangBelanja_DetailKeranjangBelanja
R_buku_detailKeranjangBelan
R_buku_kategori
Kategori
Id_Kategori (PK)
Nama_Kategori
Deskripsi
Detail_Keranjang_Belanja
Id_Keranjang_Belanja (PFK)
ISBN (PFK)
Quantity
R_buku_detailOrder
DetailOrder
ISBN (PFK)
Id_Order (PFK)
Quantity
Order_Pipeline
Order_Pipeline_Id (PK)
Tanggal_Pipeline
Pesan
Id_Order (FK)
aspnet_Profile
PropertyNames
PropertyValuesString
PropertyValuesBinary
LastUpdatedDate
UserId (PFK)
1
R_orders_detailOrder
R_orders_orderPipeline
Orders
Id_Order (PK)
UserId (FK)
Tanggal_Order
Tanggal_Pengiriman
Status_Order
Data
indek
UserId (FK)
attribut
value
R_aspnetUsers_aspnetProfile
R_orders_aspnetUsers
R_data_aspnetUsers
aspnet_Users
UserId (PK)
UserName
LoweredUserName
MobileAlias
IsAnonymous
LastActivityDate
ApplicationId (FK)
centroidGrup
UserId (PFK)
Centroid (FK)
1
R_aspnetUsers_centroidGrup
R_aspnetUsers_aspnetUsersInRoles
aspnet_UsersInRoles
UserId (PFK)
RoleId (PFK)
R_centroid_centroidGrup
R_aspnetUsers_aspnetMembership
R_aspnetApplications_aspenetUsers
1
Centroid
Centroid (PK)
Keterangan
aspnet_Membership
Password
PasswordFormat
PasswordSalt
MobilePIN
Email
LoweredEmail
PasswordQuestion
PasswordAnswer
IsApproved
IsLockedOut
CreateDate
LastLoginDate
LastPasswordChangedDate
LastLockoutDate
FailedPasswordAttemptCount
FailedPasswordAttemptWindowStart
FailedPasswordAnswerAttemptCount
FailedPasswordAnswerAttemptWindowStart
Comment
UserId (PFK)
ApplicationId (FK)
aspnet_Applications
ApplicationName
LoweredApplicationName
ApplicationId (PK)
Description
R_aspnetRoles_aspnetUsersInRoles
R_centroid_centroidInterpretasi
aspnetApplications_aspnetMembership
R_aspnetRoles_aspenetUsers
centroidInterpretasi
Attribut (PK)
value
Centroid (PFK)
aspnet_Roles
RoleId (PK)
RoleName
LoweredRoleName
Description
ApplicationId (FK)
[1,2]

DPPL

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

PEMBANGUNAN APLIKASI *CUSTOMER RELATIONSHIP MANAGEMENT (CRM)* PADA TOKO BUKU ONLINE DENGAN KLASTERISASI (MERISKA)

Dipersiapkan oleh

Pherry Chandra 03.07.03782

Program Studi Teknik Informatika

Universitas Atma Jaya Yogyakarta

Jalan Babarsari 43 Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	1. Revisi pada design class 2. Revisi Realisasi Use Case
B	Penambahan deskripsi pada perancangan antarmuka.
C	
D	
E	
F	

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh	PHERRY							
Diperiksa oleh	PM ERN							
Disetujui oleh	PM ERN							

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi
7 – 83	A		
94 – 118	B		

Daftar Isi

1 Pendahuluan 6
1.1 Tujuan 6
1.2 Lingkup Dokumen 6
1.3 Definisi, Akronim, dan Singkatan 6
1.4 Referensi 6
1.5 Deskripsi Umum Dokumen 6
2 Deskripsi Perancangan Arsitektural 7
2.1 Deployment Diagram 7
2.2 Design Class 7
2.3 Realisasi Use Case 47
2.3.1 Use Case : Login 47
2.3.2 Use Case : Pengubahan Profil Customer 48
2.3.3 Use Case : Pencarian Catalog 49
2.3.4 Use Case : Pengisian Keranjang Belanja 50
2.3.5 Use Case : Order 56
2.3.6 Use Case : Pengelolaan Catalog 58
2.3.7 Use Case : Pengelolaan Keranjang Belanja 70
2.3.8 Use Case : Pengeloaan Order 72
2.3.9 Use Case : Pengeloaan Klaster 74
2.3.10 Use Case : Notifikasi Pembayaran 76
2.3.11 Use Case : Tampil Informasi 77
2.3.12 Use Case : Pengelolaan Email 78
2.3.13 Use Case : Pengelolaan Stok 82
3 Deskripsi Perancangan Persistent Data 84
3.1 Basis Data 84
3.1.1 Tabel Buku 87
3.1.2 Tabel Kategori 87
3.1.3 Tabel Stok 87
3.1.4 Tabel Keranjang Belanja 87
3.1.5 Tabel Detail Keranjang Belanja 88
3.1.6 Tabel Orders 88
3.1.7 Tabel Detail Order 89
3.1.8 Tabel Order Pipeline 89
3.1.9 Tabel Data 89
3.1.10 Tabel Centroid Interpretasi 90
3.1.11 Tabel Centroid 90
3.1.12 Tabel Centroid Grup 90
3.1.13 Tabel aspnet_Profile 91
3.1.14 Tabel aspnet_Membership 91
3.1.15 Tabel aspnet_Applications 92
3.1.16 Tabel aspnet_Roles 92
3.1.17 Tabel aspnet_Users 92
3.1.18 Tabel aspnet_UsersInRoles 93
4 Deskripsi Perancangan Antarmuka 94
4.1 Use Case : Login 94
4.2 Use Case : Registrasi 95
4.3 Use Case : Pengubahan Profil Customer 96
4.4 Use Case : Pencarian Catalog 97
4.5 Use Case : Pengisian Keranjang Belanja 98
4.6 Use Case : Order 103
4.7 Use Case : Email Notifikasi 104
4.8 Use Case : Pengelolaan Catalog 105

4.9 Use Case : Pengelolaan Keranjang Belanja 111
4.10 Use Case : Pengelolaan Order 112
4.11 Use Case : Pengelolaan Klaster 114
4.12 Use Case : Notifikasi Pembayaran 115
4.13 Use Case : Tampil Informasi 116
4.14 Use Case : Pengelolaan Email 117
4.15 Use Case : Pengelolaan Stok 118

1 Pendahuluan

1.1 Tujuan

Dokumen DPPL ini dibuat untuk menyediakan deskripsi lengkap mengenai desain aplikasi *customer relationship management klaster* (MERISKA) pada toko buku online). Dokumen ini ditujukan untuk pembuat *e-commerce system* yang berfokus kepada kebutuhan *customer*.

1.2 Lingkup Dokumen

Dokumen DPPL ini menyediakan deskripsi lengkap perancangan perangkat lunak untuk MERISKA. Perancangan ini merupakan arsitektur sistem yang dijelaskan melalui perancangan class/modul, detail operasi apa yang akan dilakukan oleh masing-masing class/modul, dan layout database. Metodologi yang digunakan dalam perancangan adalah metode USDP (Unified Software Development Process) dari Rational Software

1.3 Definisi, Akronim, dan Singkatan

1.4 Referensi

Referensi yang digunakan dalam pembuatan dokumen ini adalah

- Wendy Boggs, Michael Boggs. *Mastering UML With Rational Rose.* 2002.

1.5 Deskripsi Umum Dokumen

Dokumen ini terdiri dari empat bab. Bab pertama adalah **Pendahuluan**, yang berisi deskripsi dokumen. Bab kedua adalah **Deskripsi Perancangan Arsitektural**, yang berisi deskripsi arsitektur sistem. Bab ketiga adalah **Deskripsi Perancangan Persistent Data**, yang berisi deskripsi data-data yang akan disimpan pada persistent storage. Bab keempat adalah **Deskripsi Perancangan Antarmuka**, yang berisi deskripsi rancangan GUI yang digunakan sistem untuk berinteraksi dengan user.

2 Deskripsi Perancangan Arsitektural

2.1 Deployment Diagram

2.2 Design Class

2.2.1 Pengantar

Untuk kelas-kelas yang berasal dari framework .Net digunakan nama kelas dengan package yang lengkap misalnya System.Web.Security.

Stereotype yang digunakan dalam design class adalah:

- << boundary >>

  Boundary class merupakan class yang berfungsi untuk menghubungkan sistem dengan user di luar sistem.

- << control >>

  Control class adalah suatu class yang objek-nya melakukan interaksi antar sekelompok objek lain. Control class biasanya memiliki karakteristik yang spesifik untuk satu use case, dan objek class ini biasanya hanya aktif pada realisasi use case.

- << entity >>

  Entity class adalah class yang bersifat pasif, dalam arti class tersebut tidak memulai interaksi dengan class lain. Entity class ini biasanya merepresentasikan suatu objek yang disimpan dalam persistent storage.

2.2.2 Class MERISKA.AksesData.AksesDataUmum

<< Control >> AksesDataUmum
+ExecuteSelectCommand (command : DbCommand) : DataTable +CreateCommand () : Dbcommand +ExecuteNonQuery (command : DbCommand) : int +ExecuteScalar (command : DbCommand) : string

Gambar 2.1 Class MERISKA.AkssData.AksesDataUmum

Deskripsi

Class yang berperan sebagai control class untuk aksi yang berhubungan dengan akses ke basis data yang bersifat umum. Aksi yang berhubungan dengan class ini yaitu:

- Menciptakan objek command (CreateCommand).

- Mengeksekusi query yang berupa command dan menampilkannya dalam bentuk Data Table (ExecuteSelectCommand).
- Mengeksekusi query yang berupa command dan menghasilkan result tunggal yang bertipe integer (ExecuteNonQuery).
- Mengeksekusi query yang berupa command dan menghasilkan result tunggal yang bertipe string (ExecuteScalar).

Atribut

-

Method

- *+ ExecuteSelectCommand (command : DbCommand) : DataTable*
  Digunakan untuk menghasilkan data table dari suatu query.
  Parameters:
  command - berisi query untuk menghasilkan suatu result data table.
  Return:
  Data Table - merupakan result dari query yang telah ter-eksekusi.
- *+ CreateCommand () : Dbcommand*
  Merupakan fungsi untuk menciptakan command.
  Return:
  Dbcommand - Hasil konfigurasi dari provider name, connection string, connection dan type command.
- *+ ExecuteNonQuery (command : DbCommand) : int*
  Digunakan untuk menghasilkan result tunggal dari suatu query yang bertipe integer.
  Parameters:
  command - berisi query untuk menghasilkan suatu result tunggal.
  Return:
  Int - merupakan result tunggal yang bertipe integer.
- *+ ExecuteScalar (command : DbCommand) : string*
  Digunakan untuk menghasilkan result tunggal dari suatu query yang bertipe string.
  Parameters:
  command - berisi query untuk menghasilkan suatu result tunggal.
  Return:
  Int - merupakan result tunggal yang bertipe string.

2.2.3 Class MERISKA.AksesData.AksesCatalog

<< Entity >> AkscsCatalog
+getKategori (): DataTable +getKategoriParamNamaKategori (NamaKategori: string): DataTable +getIdKategori (nama: string): string +UpdateKategori (IdKategori: string, NamaKategori: string, Deskripsi: string): bool +DeleteKategori (IdKategori: string): bool +AddKategori (IdKategori: string, NamaKategori: string, Deskripsi: string): bool +AddBuku (isbn: string, judul: string, deskripsi: string, penulis: string, penerbit: string, harga: string, diskon: string, image: string, idKategori: string): bool +DeleteBuku (isbn: string): bool

+UpdateBuku (isbn: string, judul: string, deskripsi: string, penulis: string, penerbit: string, harga: string, diskon: string, statusBaru: string, image: string): bool
+getBuku (): DataTable
+getBukuParamIdKategori (idKategori: string): DataTable
+getBukuParamJudul (judul: string): DataTable
+getBukuParamISBN (isbn: string): DataTable
+UpdateIdKategoriBuku (isbn: string, kategori: string): bool
+ListBuku (pageNumber: string, out jmlTotalPage: int): DataTable
+ListBukuParamIdKategori (idKategori: string, pageNumber: string, out jmlTotalPage: int) : DataTable
+ListBuku Baru(pageNumber: string): DataTable
+ListBuku BaruKlaster(pageNumber: string, username: string): DataTable
+ListSatuBuku Baru(pageNumber: string): DataTable
+ListSatuBuku BaruKlaster(pageNumber: string, username: string): DataTable
+ListBukuTerlaris(pageNumber: string): DataTable
+ListBukuTerlarisKlaster(pageNumber: string, username: string): DataTable
+ListSatuBukuTerlaris(pageNumber: string): DataTable
+ListSatuBukuTerlarisKlaster(pageNumber: string, username: string): DataTable
+ListSatuBukuJarangDiBeli(pageNumber: string): DataTable
+jumTotPageBukuBaru (): int
+jumTotPageBukuBaruKlaster (username: string): int
+jumTotPageSatuBukuBaru (): int
+jumTotPageSatuBukuBaruKlaster (username: string): int
+jumTotPageSatuBukuJarangDiBeli (): int
+getKategoriParamIdKategori (idKategori: string): DetailKategori
+Search (pageNumber: string, out jmlTotalPage, key: string, text: string): DataTable
+ListBukuKlaster (pageNumber: string, out jmlTotalPage: int, username: string): DataTable

Gambar 2.2 Class MERISKA.AksesData.AksesCatalog

Deskripsi

Class yang berperan sebagai Entity class untuk aksi yang berhubungan dengan akses catalog ke basis data. Aksi yang berhubungan dengan class ini yaitu:

- Get kategori dan id kategori.
- Get kategori berdasarkan parameter tertentu (id kategori dan nama).
- Tambah kategori, update kategori dan hapus kategori.
- Get buku, tambah buku, update buku, hapus buku dan cari buku.
- Get buku bedasarkan parameter tertentu (id kategori, isbn, judul).
- Get list buku, list buku klaster, list buku baru terklaster,list buku terlaris, list buku jarang dibeli, dan list buku berdasarkan parameter.
- Get jumlah total page (buku baru, buku baru klaster, buku jarang dibeli dan satu buku baru).

Atribut

-

Method

- *+getKategori () : DataTable*

  Digunakan untuk mengambil semua kategori dalam basis data.

  Return:

  Semua kategori yang tersimpan dalam basis data.
- *+getKategoriParamNamaKategori (NamaKategori : string) : DataTable*

Digunakan untuk mengambil kategori dalam basis data yang sesuai dengan parameter *NamaKategori*.

Parameters:

NamaKategori – merupakan parameter untuk filtering kategori yang akan diambil.

Return:

Semua kategori yang tersimpan dalam basis data yang sesuai dengan parameter *NamaKategori*.

- *+ getIdKategori (nama : string) : string*

  Digunakan untuk mengambil Id kategori yang sesuai dengan parameter *nama*.

  Parameters:

  nama – sebagai parameter filtering untuk mendapatkan id kategori.

  Return:

  Id kategori yang sesuai dengan parameter *nama*.

- *+ UpdateKategori (IdKategori : string, NamaKategori : string, Deskripsi : string) : bool*

  Mengubah data kategori berdasarkan id kategori tertentu .

  Parameters:

  Id kategori - id dari data kategori yang akan diubah, NamaKategori - nama kategori yang akan diubah, Deskripsi - id deskripsi kategori yang akan diubah

  Return:

  Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ DeleteKategori (IdKategori : string) : bool*

  Menghapus data kategori berdasarkan id kategori tertentu .

  Parameters:

  Id kategori - id dari data kategori yang akan dihapus.

  Return:

  Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ AddKategori (IdKategori : string, NamaKategori : string, Deskripsi : string) : bool*

  Menambah data kategori baru

  Parameters:

  Id kategori, NamaKategori, Deskripsi - merupakan data kategori baru yang akan ditambahkan.

  Return:

  Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ AddBuku (isbn : string, judul : string, deskripsi : string, penulis : string, penerbit : string, harga : string, diskon : string, image: string, idKategori : string) : bool*

  Menambah data kategori baru

  Parameters:

  Isbn, judul, deskripsi, penulis, penerbit, harga, diskon, image, idkategori - merupakan data buku baru yang akan ditambahkan.

  Return:

  Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ DeleteBuku (isbn : string) : bool*

  Menghapus data buku berdasarkan isbn tertentu .

Parameters:
isbn - merupakan isbn dari data buku yang akan dihapus.
Return:
Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ UpdateBuku (isbn : string, judul : string, deskripsi : string, penulis : string, penerbit : string, harga : string, diskon : string, statusBaru: string, image: string) : bool*

Mengubah data buku berdasarkan isbn tertentu .
Parameters:
Isbn - merupakan isbn dari buku yang akan diubah.
judul, deskripsi, penulis, penerbit, harga, diskon, image, statusBaru - merupakan data – data buku yang akan diubah.
Return:
Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ getBuku () : DataTable*

Digunakan untuk mengambil semua buku dalam basis data.
Return:
Semua buku yang tersimpan dalam basis data.

- *+ getBukuParamIdKategori (idKategori : string) : DataTable*

Digunakan untuk mengambil semua buku dalam basis data yang sesuai dengan id kategori tertentu.
Parameters:
IdKategori - merupakan parameter untuk filtering buku yang akan diambil.
Return:
Semua buku yang tersimpan dalam basis data yang sesuai dengan parameter *IdKategori*.

- *+ getBukuParamJudul (judul : string) : DataTable*

Digunakan untuk mengambil semua buku dalam basis data yang sesuai dengan judul tertentu
Parameters:
judul - merupakan parameter untuk filtering buku yang akan diambil.
Return:
Semua buku yang tersimpan dalam basis data yang sesuai dengan parameter *judul*.

- *+ getBukuParamISBN (isbn : string) : DataTable*

Digunakan untuk mengambil semua buku dalam basis data yang sesuai dengan isbn tertentu
Parameters:
isbn - merupakan parameter untuk filtering buku yang akan diambil.
Return:
Semua buku yang tersimpan dalam basis data yang sesuai dengan parameter *isbn*.

- *+ UpdateIdKategoriBuku (isbn : string, kategori: string) : bool*

Digunakan untuk mengubah id kategori buku dalam basis data.
Parameters:
isbn - merupakan parameter untuk filtering buku yang akan diubah.

kategori - merupakan parameter untuk id kategori pengganti.

Return:

Buku yang tersimpan dalam basis data yang sesuai dengan parameter *isbn*.

- *+ ListBuku (pageNumber : string, out jmlTotalPage : int)*

  Digunakan untuk mengambil data buku / list buku dalam basis.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*. jmlTotalPage - merupakan reprensentasi jumlah total page yang dihasilkan dari keseluruhan jumlah buku.

  Return:

  List buku.

- + ListBukuParamIdKategori (idKategori:string, pageNumber:string, out jmlTotalPage:int)

  Digunakan untuk mengambil data buku / list buku dalam basis data yang sesuai dengan id kategori tertentu.

  Parameters:

  pageNumber - merupakan representasi untuk paging, IdKategori - merupakan parameter untuk filtering buku id kategori-nya. jmlTotalPage - merupakan reprensentasi jumlah total page yang dihasilkan dari keseluruhan jumlah buku.

  Return:

  List buku berdasarkan id kategori tertentu.

- *+ ListBukuBaru (pageNumber : string)*

  Digunakan untuk mengambil data buku / list buku baru secara random dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  Return:

  List buku baru.

- *+ ListBukuBaruKlaster (pageNumber : string, username: string)*

  Digunakan untuk mengambil data buku / list buku baru yang telah ter-klaster secara random dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  username - merupakan filter untuk menampilkan list buku yang telah ter-klaster..

  Return:

  List buku baru klaster.

- *+ ListSatuBukuBaru (pageNumber : string)*

  Digunakan untuk mengambil satu buku baru secara random dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  Return:

  List satu buku baru.

- *+ ListSatuBukuBaruKlaster (pageNumber : string, username: string)*

Digunakan untuk mengambil satu buku baru yang telah ter-klaster secara random dalam basis data.

Parameters:

pageNumber - merupakan representasi untuk *paging*.

username - merupakan filter untuk menampilkan list satu buku baru yang telah ter-klaster..

Return:

List satu buku baru klaster.

- *+ ListBukuTerlaris (pageNumber : string)*

  Digunakan untuk mengambil list buku terlaris dalam dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  Return:

  List buku terlaris.

- *+ ListBukuTerlarisKlaster (pageNumber : string, username: string)*

  Digunakan untuk mengambil list buku terlaris yang telah ter-klaster dalam dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  username - merupakan filter untuk menampilkan list buku terlaris yang telah ter-klaster..

  Return:

  List buku terlaris klaster.

- *+ ListSatuBukuTerlaris (pageNumber : string)*

  Digunakan untuk mengambil satu buku terlaris dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  Return:

  List satu buku baru.

- *+ ListSatuBukuTerlarisKlaster (pageNumber : string, username: string)*

  Digunakan untuk mengambil satu buku terlaris yang telah ter-klaster dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  username - merupakan filter untuk menampilkan list satu buku terlaris yang telah ter-klaster..

  Return:

  List satu buku terlaris klaster.

- *+ ListSatuBukuJarangDiBeli (pageNumber : string)*

  Digunakan untuk mengambil satu buku jarang dibeli dalam basis data.

  Parameters:

  pageNumber - merupakan representasi untuk *paging*.

  Return:

  List satu buku jarang dibeli.

- *+ jumTotPageBukuBaru () : int*

  Fungsi untuk mendapatkan jumlah total page buku baru.

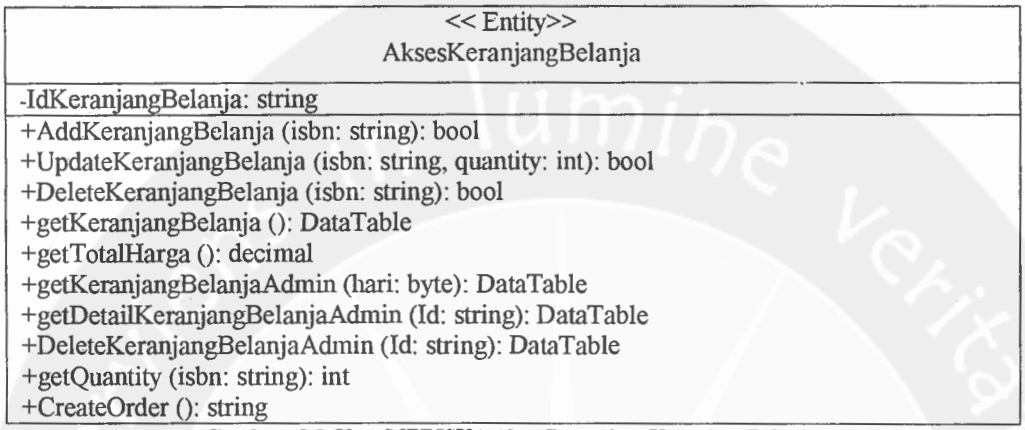
  Return:

Jumlah total page buku baru.

- *+ jumTotPageBukuBaruKlaster (username: string) : int*
  Fungsi untuk mendapatkan jumlah total page buku baru yang telah ter-klaster.
  Parameters:
  username - merupakan filter untuk menampilkan total page buku baru yang telah ter-klaster.
  Return:
  Jumlah total page buku baru yang telah ter-klaster.
- *+ jumTotPageSatuBukuBaru () : int*
  Fungsi untuk mendapatkan jumlah total page satu buku baru.
  Return:
  Jumlah total page satu buku baru.
- *+ jumTotPageSatuBukuBaruKlaster () : int*
  Fungsi untuk mendapatkan jumlah total page satu buku baru yang telah ter-klaster.
  Return:
  Jumlah total page satu buku baru yang telah ter-klaster.
- *+ jumTotPageSatuBukuJarangDiBeli () : int*
  Fungsi untuk mendapatkan jumlah total page satu buku jarang dibeli .
  Return:
  Jumlah total page satu buku jarang dibeli.
- *+ getKategoriParamIdKategori (idKategori : string) : DetailKategori*
  Digunakan untuk mendapatkan nama kategori dalam basis data yang sesuai dengan id-nya.
  Parameters:
  idKategori - merupakan parameter untuk mendapatkan nama kategori.
  Return:
  Nama kategori yang sesuai dengan id-nya.
- *+ Search (pageNumber : string, out jmlTotalPage, key : string, text : string) : DataTable*
  Fungsi untuk mencari buku berdasarkan kriteria tertentu (key-nya) dan text-nya.
  Parameters:
  pageNumber - merupakan representasi untuk paging, IdKategori - merupakan parameter untuk filtering buku id kategori-nya. jmlTotalPage - merupakan reprensentasi jumlah total page yang dihasilkan dari keseluruhan jumlah buku, key - merupakan atribut pencarian, text - merupakan inputan pencarian.
  Return:
  List buku yang dicari.
- *+ ListBukuKlaster (pageNumber : string, out jmlTotalPage : int, username: string)*
  Digunakan untuk mengambil data buku / list buku yang telah ter-klaster dalam basis.
  Parameters:

pageNumber - merupakan representasi untuk *paging*. jmlTotalPage - merupakan reprensentasi jumlah total page yang dihasilkan dari keseluruhan jumlah buku.
username - merupakan filter untuk menampilkan list buku yang telah ter-klaster.
Return:
List buku yang telah ter-klaster.

2.2.4 Class MERISKA.AksesData.AksesKeranjangBelanja



Gambar 2.3 Class MERISKA.AksesData.AksesKeranjangBelanja

Deskripsi

Class yang berperan sebagai entity class untuk aksi yang berhubungan dengan akses keranjang belanja ke basis data. Aksi yang berhubungan dengan class ini yaitu:

- Get keranjang belanja dan keranjang belanja admin.
- Tambah item di keranjang belanja
- Update item di keranjang belanja.
- Hapus item di keranjang belanja dan keranjang belanja admin.
- Get total harga belanja.
- Get detail keranjang belanja admin.
- Get Quantity.
- Create Order.

Atribut

- *+ IdKeranjangBelanja : string*
  Merepresentasikan Id keranjang belanja.

Method

- *+ AddKeranjangBelanja (isbn : string) : bool*
  Menambah item dalam keranjang belanja.
  Parameters:
  isbn - merupakan isbn dari buku yang akan dimasukan ke keranjang belanja.
  Return:
  Bernilai true jika item berhasil ditambahkan ke keranjang belanja / ada baris yang berubah dan false bila sebaliknya.
- *+ UpdateKeranjangBelanja (isbn : string, quantity : int) : bool*

Mengubah quantity buku-buku yang terdapat di keranjang belanja berdasarkan isbn-nya masing-masing.

Parameters:

isbn - isbn dari buku yang akan diubah quantity-nya, Quantity - jumlah dari buku yang terdapat di keranjang belanja.

Return:

Bernilai true jika ada baris yang berubah.

- *+ DeleteKeranjangBelanja (isbn: string) : bool*

  Menghapus buku yang berada dikeranjang belanja berdasarkan isbn-nya .

  Parameters:

  isbn - merupakan isbn dari data buku yang akan dihapus dari keranjang belanja.

  Return:

  Bernilai true jika ada baris yang berubah.

- *+ getKeranjangBelanja () : DataTable*

  Digunakan untuk mengambil data keranjang belanja dalam basis data.

  Return:

  List buku dalam keranjang belanja.

- *+ getTotalHarga () : DataTable*

  Digunakan untuk mengambil total harga item di keranjang belanja.

  Return:

  List total harga item di keranjang belanja.

- *+ getKeranjangBelanjaAdmin (hari: byte) : DataTable*

  Digunakan untuk mengambil list-list keranjang belanja dalam basis data berdasarkan limit hari.

  Parameters:

  hari - merupakan parameter filtering untuk menentukan limit hari dari list keranjang belanja.

  Return:

  List keranjang belanja berdasarkan limit hari.

- *+ getDetailKeranjangBelanjaAdmin (Id: string) : DataTable*

  Digunakan untuk mengambil detail keranjang belanja dalam basis data berdasarkan Id keranjang belanja.

  Parameters:

  Id - merupakan parameter filtering untuk mengambil detail keranjang belanja berdasarkan Id-nya.

  Return:

  Detail keranjang belanja berdasarkan Id-nya.

- *+ DeleteKeranjangBelanjaAdmin (Id: string) : bool*

  Menghapus list keranjang belanja berdasarkan id-nya .

  Parameters:

  id - merupakan Id dari list keranjang belanja yang akan dihapus dari basis data.

  Return:

  Bernilai true jika ada baris yang berubah.

- *+ getQuantity (isbn: string) : int*

Merupakan fungsi untuk mendapatkan quantity buku berdasarkan isbn tertentu.

Parameters:

isbn - merupakan isbn dari data buku yang akan didapatkan quantity-nya.

Return:

Quantity buku.

- *+ CreateOrder () : string*

  Merupakan fungsi untuk menciptakan order.

  Return:

  Id Order baru.

2.2.5 Class MERISKA.AksesData.AksesOrder

<< Entity >> AksesOrder
+GetOrderDetails (idOrder: string): DataTable +ListGetOrderDetails (orderId: string): List<AksesOrderDetailInfo> +getTotalHargaOrder (idOrder: string): decimal +getStatusOrder (idOrder: string): int +UpdateTglPengiriman(idOrder: string) +OrderCancel(idOrder: string) +getOrder(IdOrder: string): AksesOrderInfo +CreateOrderPipeline (IdOrder: int, pesan: string) +UpdateOrderStatus(IdOrder: int, status: int) +SetTanggalPengiriman(IdOrder: int) +GetOrderPipeline(IdOrder: int): List<IdOrder: string> +ConvertDataTableToOrders(table: DataTable):List <aksesOrderInfo> +GetOrdersByCustomer (customerID: string): List<AksesOrderInfo> +GetOrdersByDate (startDate: string, endDate: string): List<AksesOrderInfo> +UpdateOrder(orderID: int, newTglOrder: string, newTglKirim: string, newStatus: int)

Gambar 2.4 Class MERISKA.AksesData.AksesOrder

Deskripsi

Class yang berperan sebagai control class untuk aksi yang berhubungan dengan akses order ke basis data secara umum. Aksi yang berhubungan dengan class ini yaitu:

- Get order detail berupa data table.
- Get total harga order
- Get status order.
- Update tanggal pengiriman.
- Cancel Order.

Atribut

-

Method

- *+ GetOrderDetail (idOrder: string) : DataTable*

  Digunakan untuk mengambil detail order dalam basis data berdasarkan Id order.

  Parameters:

  idOrder - merupakan parameter filtering untuk mengambil detail order berdasarkan Id-nya.

Return:
Detail order berdasarkan Id-nya.

- *+ ListGetOrderDetaisl (orderId: string) : List<AksesOrderDetailInfo>*
  Digunakan untuk mengambil list detail order dalam basis data berdasarkan Id order.
  Parameters:
  idOrder - merupakan parameter filtering untuk mengambil list detail order berdasarkan Id-nya.
  Return:
  List detail order berdasarkan Id-nya.
- *+ GetTotalHargaOrder (idOrder: string) : decimal*
  Digunakan untuk mengambil total harga order berdasarkan Id order.
  Parameters:
  idOrder - merupakan parameter filtering untuk mengambil total harga order berdasarkan Id-nya.
  Return:
  Total harga order berdasarkan Id-nya.
- *+ GetStatusOrder (idOrder: string) : int*
  Digunakan untuk mengambil status order berdasarkan Id order.
  Parameters:
  idOrder - merupakan parameter filtering untuk mengambil status order berdasarkan Id-nya.
  Return:
  Status order berdasarkan Id-nya.
- *+ UpdateTglPengiriman (idOrder: string)*
  Mengubah tanggal pengiriman order berdasarkan id order-nya masing-masing.
  Parameters:
  idOrder – id order dari data order yang akan diubah tgl pengiriman-nya.
  Return:
  -
- *+ OrderCancel (idOrder: string)*
  Mengubah status order menjadi status cancel.
  Parameters:
  idOrder – id order dari data order yang akan diubah statusnya menjadi cancel.
  Return:
  -
- *+ GetOrder (IdOrder: string) : AksesOrderInfo*
  Digunakan untuk mengambil order dalam basis data berdasarkan Id order.
  Parameters:
  idOrder - merupakan parameter filtering untuk mengambil order berdasarkan Id-nya.
  Return:
  Order berdasarkan Id-nya.
- *+ CreateOrderPipeline (order: AksesOrder, IdOrder: int, pesan: string)*

Merupakan fungsi untuk menciptakan order pipeline.

Parameters:

IdOrder, pesan - merupakan parameter untuk menciptakan order pipeline.

Return:

- *+ UpdateOrderStatus (IdOrder: int, status: int)*

  Mengubah status order berdasarkan id order-nya masing-masing.

  Parameters:

  idOrder – id order dari data order yang akan diubah status order-nya.

  Return:

  -

- *+ SetTglPengiriman (idOrder: int)*

  Mengeset tanggal pengiriman order berdasarkan id order-nya masing-masing.

  Parameters:

  idOrder – id order dari data order yang akan diset tgl pengiriman-nya.

  Return:

  -

- *+ GetOrderPipeline (IdOrder: string) : List<OrderPipeline>*

  Digunakan untuk mengambil list order pipeline dalam basis data berdasarkan Id order.

  Parameters:

  idOrder - merupakan parameter filtering untuk mengambil list order pipeline berdasarkan id order.

  Return:

  List order pipeline berdasarkan Id order.

- *+ ConvertDataTableToOrders (table: DataTable) : List<AksesOrderInfo>*

  Digunakan untuk meng-konversi order yang berbentuk data table menjadi list.

  Parameters:

  table - merupakan parameter inputan data table yang akan diubah menjadi list.

  Return:

  List order.

- *+ GetOrderByCustomer (customerID: string) : List<AksesOrderInfo>*

  Digunakan untuk mengambil list order berdasarkan customer.

  Parameters:

  customerID - merupakan parameter filtering berdasarkan id customer untuk list order yang akan diambil.

  Return:

  List order berdasarkan customer.

- *+ GetOrderByDate (startDate: string, endDate: string) : List<AksesOrderInfo>*

  Digunakan untuk mengambil list order berdasarkan tanggal.

  Parameters:

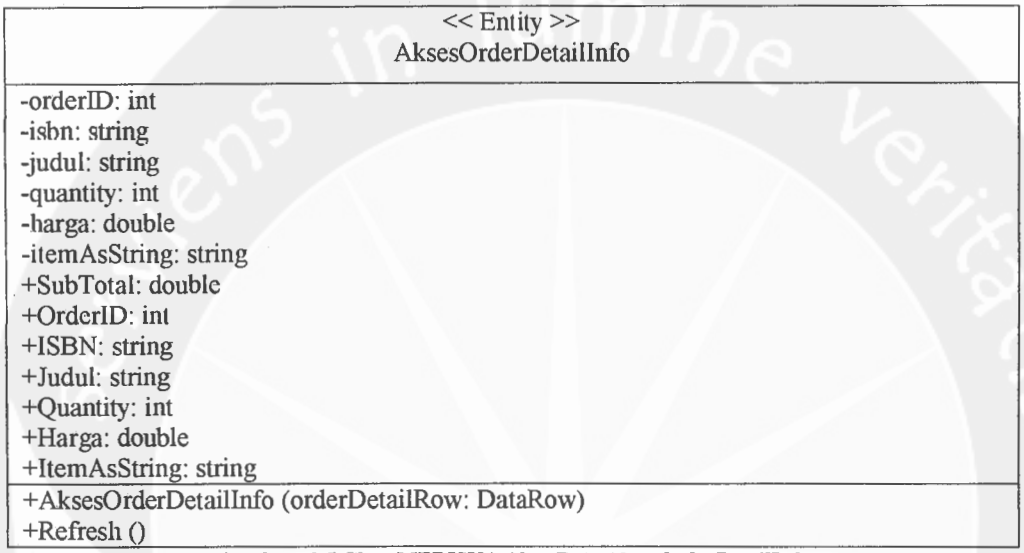
  startDate, endDate - merupakan parameter filtering berdasarkan tanggal untuk list order yang akan diambil.

  Return:

  List order berdasarkan tanggal.

- *+ UpdateOrder (orderID: int, newTglOrder: string, newTglKirim: string, newStatus: int)*
  Mengubah beberapa attribute order berdasarkan id order-nya masing-masing.
  Parameters:
  idOrder – id order dari data order yang akan diubah.
  newTglOrder, newTglKirim, newStatus – merupakan parameter inputan untuk attribute-attrbut yang akan diubah.
  Return:
  -

2.2.6 Class MERISKA.AksesData.AksesOrderDetailInfo



Gambar 2.5 Class MERISKA.AksesData.AksesOrderDetailInfo

Deskripsi

Class yang berperan sebagai entity class. Mendeskripsikan order detail info.

Atribut

- *- orderID : int*
  Merepresentasikan id order.
- *- isbn : string*
  Merepresentasikan isbn dari buku yang diorder.
- *- judul : string*
  Merepresentasikan judul dari buku yang diorder.
- *- quantity : int*
  Merepresentasikan quantity dari buku yang diorder.
- *- harga : int*
  Merepresentasikan harga dari buku yang diorder.
- *- itemAsString : string*
  Merepresentasikan kumpulan informasi dari data-data order sebagai suatu string.
- *+ SubTotal : double*
  Merepresentasikan sub total dari setiap buku yang diorder.
- *+ OrderID : int*

Attribut untuk set dan get id order.

- *+ ISBN : string*
  Attribut untuk set dan get isbn dari buku yang diorder.
- *+ Judul : string*
  Attribut untuk set dan get judul dari buku yang diorder.
- *+ Quantity : int*
  Attribut untuk set dan get quantity dari buku yang diorder.
- *+ harga : int*
  Attribut untuk set dan get harga dari buku yang diorder.
- *+ itemAsString : string*
  Attribut untuk set dan get informasi dari data-data order.

Method

- *+ AksesOrderDetailInfo (orderDetailRow: DataRow)*
  Buat instance baru *AksesOrderDetailInfo* dengan attribut orderDetailRow.
- *+ refresh*
  Method untuk menampilkan *itemAsString*.

2.2.7 Class MERISKA.AksesData.AksesOrderInfo

<< Entity >> AksesOrderInfo
-IdOrder: int -TanggalOrder: string -TanggalPengiriman: string -StatusOrder: int -customer: MembershipUser -orderDetail: List<AksesOrderDetailInfo> -orderPipeline: List<OrderPipeline> +CustomerProfile: ProfileCommon +TotalHarga: double +OrderAsString: string +AlamatCustomerAsString: string +Status_Order: string +Id_Order: int +Tanggal_Order: string +Tanggal_Pengiriman: string +Customer: MembershipUser +OrderPipeline: List<OrderPipeline> +OrderDetail: List<AksesOrderDetailInfo>
+AksesOrderLibInfo (orderDetailRow: DataRow) +Refresh () +UpdateStatus (status: int) +SetTanggalPengiriman ()

Gambar 26 Class MERISKA.AksesData.AksesOrderInfo

Deskripsi

Class yang berperan sebagai entity class. Mendeskripsikan order info.

Atribut

- *- IdOrder : int*
  Merepresentasikan id order.
- *-TanggalOrder : string*
  Merepresentasikan tanggal order.
- *- TanggalPengiriman : string*

Merepresentasikan tanggal pengiriman.

- - Status*Order : int*
  Merepresentasikan status order.
- - customer *: MembershipUser*
  Merepresentasikan customer.
- - orderDetail *: List<AksesOrderDetailInfo>*
  Merepresentasikan detail order.
- - orderPipeline *: List<OrderPipeline>*
  Merepresentasikan order pipeline.
- + CustomerProfile *: ProfileCommon*
  Merepresentasikan customer profile.
- + TotalHarga *: double*
  Merepresentasikan total harga.
- + OrderAsString *: string*
  Merepresentasikan keseluruhan informasi order sebagai suatu string.
- + AlamatCustomerAsString *: string*
  Merepresentasikan alamat customer sebagai suatu string.
- + Status_*Order : int*
  Attribut untuk set dan get status order.
- + Id_*Order: int*
  Attribut untuk set dan get id order.
- + Tanggal_*Order : string*
  Attribut untuk set dan get id tanggal order.
- + Tanggal_*Pengiriman : string*
  Attribut untuk set dan get id tanggal pengiriman.
- + Customer *: MembershipUser*
  Attribut untuk set dan get id customer.
- + OrderPipeline *: List<OrderPipeline>*
  Attribut untuk set dan get Order pipeline.
- + OrderDetail *: List<AksesOrderDetailInfo>*
  Attribut untuk set dan get detail order.

Method

- + *AksesOrderInfo (orderDetailRow: DataRow)*
  Buat instance baru *AksesOrderInfo* dengan attribut orderDetailRow.
- + *refresh*
  Method untuk menampilkan *OrderAsString* dan *AlamatCustomerAsString*
  List order berdasarkan tanggal.
- + *UpdateStatus (status: int)*
  Mengubah status order menjadi status yang sesuai dengan parameter inputan.
  Parameters:
  status – merupakan parameter inputan untuk attribute yang akan diubah.
  Return:
  -
- + *SetTglPengiriman ()*
  Mengeset tanggal pengiriman order.
  Parameters:

-

Return:

-

2.2.8 Class MERISKA.AksesData.OrderPipeline

<< Entity>> OrderPipeline
-OrderPipelineId: int -IdOrder: int -TanggalPipeline: DateTime -pesan: string
+OrderPipeline (orderPipeline: DataRow) +Order_Pipeline_Id: int +Id_Order: int +Tanggal_Pipeline: DateTime +Pesan: string

Gambar 27 Class MERISKA.AksesData.OrderPipeline

Deskripsi

Class yang berperan sebagai entity class. Mendeskripsikan order pipeline.

Atribut

- *- OrderPipelineId : int*
  Merepresentasikan id order pipeline.
- - IdOrder *: int*
  Merepresentasikan id order.
- - TanggalPipeline *: DateTime*
  Merepresentasikan tanggal terjadinya track order.
- -pesan *: string*
  Merepresentasikan pesan.
- *+ Order_Pipeline_Id: int*
  Attribut untuk set dan get id order pipeline.
- + Id_Order *: int*
  Attribut untuk set dan get id order.
- + Tanggal_Pipeline *: DateTime*
  Attribut untuk set dan get id tanggal pipeline.
- + Pesan *: string*
  Attribut untuk set dan get pesan.

Method

- *+ OrderPipeline (orderPipeline: DataRow)*
  Buat instance baru *OrderPipeline* dengan attribut orderPipeline.

2.2.9 Class MERISKA.AksesData.AksesKlaster

<< Entity >> AksesKlaster
+getKlaster (): DataTable +UpdateKlaster (centroid: string, keterangan: string): bool

Gambar 28 Class MERISKA.AksesData.AksesKlaster

Deskripsi

Class yang berperan sebagai entity class untuk aksi yang berhubungan dengan akses klaster ke basis data. Aksi yang berhubungan dengan class ini yaitu:

- Get klaster.
- Update klaster.

Atribut

-

Method

- *+ getKlaster () : DataTable*
  Digunakan untuk mengambil data klaster dalam basis data.
  Parameters:
  -
  Return:
  List Klaster.
- *+ UpdateKlaster (centroid: string, keterangan: string)*
  Mengubah label klaster.
  Parameters:
  keterangan – merupakan parameter inputan untuk attribute yang akan diubah.
  centroid – merupakan parameter filtering untuk klaster yang akan diubah.
  Return:
  Bernilai true jika ada baris yang berubah.

2.2.10 Class MERISKA.AksesData.AksesStok

<< Entity >> AksesStok
+AddStok (Isbn: string): bool +UpdateStok (Isbn: string, Quantity: string): bool +getStokBuku (isbn: string): int +getDataStok (): DataTable +getStokParamJudul (judul: string): DataTable +stokEmpty (isbn: string): bool

Gambar 29 Class MERISKA.AksesData.AksesStok

Deskripsi

Class yang berperan sebagai control class untuk aksi yang berhubungan dengan akses stok ke basis data. Aksi yang berhubungan dengan class ini yaitu:

- Tambah stok.
- Update stok.
- Get data stok (list stok) ,get stok buku (jumlahnya), get stok berdasarkan parameter judul (list stok)
- Cek stok empty

Atribut

-

Method

- *+ AddStok (Isbn : string) : bool*
  Menciptakan data stok baru untuk buku tertentu di basis data.

Parameters:
Isbn - merupakan parameter inputan untuk buku tertentu yang akan diciptakan data stok barunya.
Return:
Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ UpdateKategori (Isbn: string, Quantity : string) : bool*

Mengubah data stok berdasarkan isbn tertentu .
Parameters:
Isbn - isbn dari data stok yang akan diubah, Quantity – quantity stok yang diubah
Return:
Bernilai true jika ada baris yang berubah dan false bila sebaliknya.

- *+ getStokBuku (Isbn : string) : int*

Mendapatkan jumlah stok dari buku tertentu.
Parameters:
Isbn - merupakan parameter filtering dari buku tertentu
Return:
Jumlah stok buku tertentu.

- *+ getDataStok () : DataTable*

Digunakan untuk mengambil list data stok dalam basis data.
Parameters:
-
Return:
List Stok.

- *+ getStokParamJudul (judul: string) : DataTable*

Digunakan untuk mengambil list data stok dalam basis data berdasarkan judul.
Parameters:
Judul - merupakan parameter filtering untuk list stok.
Return:
List Stok berdasarkan judul.

- *+ stokEmpty (Isbn : string) : bool*

Mengecek stok dari buku tertentu kosong atau tidak.
Parameters:
Isbn - merupakan parameter inputan untuk buku tertentu yang akan dicek stok.
Return:
Bernilai true jika lebih kecil dari 1 dan false bila sebaliknya.

2.2.11 Class MERISKA.AksesData.AksesCatalogManager

<< Control >> AksesCatalogManager
+getKategori (catalog: AksesCatalog): DataTable +getKategoriParamNamaKategori (catalog: AksesCatalog, NamaKategori: string): DataTable +getIdKategori (catalog: AksesCatalog , nama: string): string +UpdateKategori (catalog: AksesCatalog , IdKategori: string, NamaKategori: string,

Deskripsi: string): bool
+DeleteKategori (catalog: AksesCatalog , IdKategori: string): bool
+AddKategori (catalog: AksesCatalog , IdKategori: string, NamaKategori: string, Deskripsi: string): bool
+AddBuku (catalog: AksesCatalog , isbn: string, judul: string, deskripsi: string, penulis: string, penerbit: string, harga: string, diskon: string, image: string, idKategori: string): bool
+DeleteBuku (catalog: AksesCatalog , isbn: string): bool
+UpdateBuku (catalog: AksesCatalog , isbn: string, judul: string, deskripsi: string, penulis: string, penerbit: string, harga: string, diskon: string, statusBaru: string, image: string): bool
+getBuku (catalog: AksesCatalog): DataTable
+getBukuParamIdKategori (catalog: AksesCatalog , idKategori: string): DataTable
+getBukuParamJudul (catalog: AksesCatalog , judul: string): DataTable
+getBukuParamISBN (catalog: AksesCatalog , isbn: string): DataTable
+UpdateIdKategoriBuku (catalog: AksesCatalog , isbn: string, kategori: string): bool
+ListBuku (catalog: AksesCatalog , pageNumber: string, out jmlTotalPage: int): DataTable
+ListBukuParamIdKategori (catalog: AksesCatalog , idKategori: string, pageNumber: string, out jmlTotalPage: int) : DataTable
+ListBuku Baru(catalog: AksesCatalog , pageNumber: string): DataTable
+ListBuku BaruKlaster(catalog: AksesCatalog , pageNumber: string, username: string): DataTable
+ListSatuBuku Baru(catalog: AksesCatalog , pageNumber: string): DataTable
+ListSatuBuku BaruKlaster(catalog: AksesCatalog , pageNumber: string, username: string): DataTable
+ListBukuTerlaris(catalog: AksesCatalog , pageNumber: string): DataTable
+ListBukuTerlarisKlaster(catalog: AksesCatalog , pageNumber: string, username: string): DataTable
+ListSatuBukuTerlaris(catalog: AksesCatalog , pageNumber: string): DataTable
+ListSatuBukuTerlarisKlaster(catalog: AksesCatalog , pageNumber: string, username: string): DataTable
+ListSatuBukuJarangDiBeli(catalog: AksesCatalog , pageNumber: string): DataTable
+jumTotPageBukuBaru (catalog: AksesCatalog): int
+jumTotPageBukuBaruKlaster (catalog: AksesCatalog , username: string): int
+jumTotPageSatuBukuBaru (catalog: AksesCatalog): int
+jumTotPageSatuBukuBaruKlaster (catalog: AksesCatalog , username: string): int
+jumTotPageSatuBukuJarangDiBeli (catalog: AksesCatalog): int
+getKategoriParamIdKategori (catalog: AksesCatalog, idKategori: string): DetailKategori
+Search (catalog: AksesCatalog , pageNumber: string, out jmlTotalPage, key: string, text: string): DataTable
+ListBukuKlaster (catalog: AksesCatalog , pageNumber: string, out jmlTotalPage: int, username: string): DataTable

Gambar 2.10 Class MERISKA.AkssData.AksesCatalogManager

Deskripsi

Class yang berperan sebagai control class untuk mengimplementasikan fungsi-fungsi pada class AksesCatalog ke userinterface:

Atribut

-

Method

- *+ getKategori (catalog: AksesCatalog) : DataTable*
  Fungsi yang mereturnkan fungsi *getKategori ()* pada AksesCatalog.
- *+ getKategoriParamNamaKategori (catalog: AksesCatalog , NamaKategori : string) : DataTable*

Fungsi yang mereturnkan fungsi *getKategoriParamNamaKategori (NamaKategori : string)* pada AksesCatalog.

- *+ getIdKategori (catalog: AksesCatalog , nama : string) : string*
  Fungsi yang mereturnkan fungsi *getIdKategori (nama : string)* pada AksesCatalog.
- *+ UpdateKategori (catalog: AksesCatalog , IdKategori : string, NamaKategori : string, Deskripsi : string) : bool*
  Fungsi yang mereturnkan fungsi *UpdateKategori (IdKategori : string, NamaKategori : string, Deskripsi : string) : bool* pada AksesCatalog.
- *+ DeleteKategori (catalog: AksesCatalog , IdKategori : string) : bool*
  Fungsi yang mereturnkan fungsi *DeleteKategori (IdKategori : string) : bool* pada AksesCatalog.
- *+ AddKategori (catalog: AksesCatalog , IdKategori : string, NamaKategori : string, Deskripsi : string) : bool*
  Fungsi yang mereturnkan fungsi *AddKategori (IdKategori : string, NamaKategori : string, Deskripsi : string) : bool* pada AksesCatalog.
- *+ AddBuku (catalog: AksesCatalog , isbn : string, judul : string, deskripsi : string, penulis : string, penerbit : string, harga : string, diskon : string, image: string, idKategori : string) : bool*
  Fungsi yang mereturnkan fungsi *AddBuku (isbn : string, judul : string, deskripsi : string, penulis : string, penerbit : string, harga : string, diskon : string, image: string, idKategori : string) : bool* pada AksesCatalog.
- *+ DeleteBuku (catalog: AksesCatalog , isbn : string) : bool*
  Fungsi yang mereturnkan fungsi *DeleteBuku (isbn : string) : bool* pada AksesCatalog.
- *+ UpdateBuku (catalog: AksesCatalog , isbn : string, judul : string, deskripsi : string, penulis : string, penerbit : string, harga : string, diskon : string, statusBaru: string, image: string) : bool*
  Fungsi yang mereturnkan fungsi *UpdateBuku (isbn : string, judul : string, deskripsi : string, penulis : string, penerbit : string, harga : string, diskon : string, statusBaru: string, image: string) : bool* pada AksesCatalog.
- *+ getBuku (catalog: AksesCatalog) : DataTable*
  Fungsi yang mereturnkan fungsi *getBuku () : DataTable* pada AksesCatalog.
- *+ getBukuParamIdKategori (catalog: AksesCatalog , idKategori : string) : DataTable*
  Fungsi yang mereturnkan fungsi *getBukuParamIdKategori (idKategori : string) : DataTable* pada AksesCatalog.
- *+ getBukuParamJudul (catalog: AksesCatalog , judul : string) : DataTable*
- Fungsi yang mereturnkan fungsi *getBukuParamJudul (judul : string) : DataTable* pada AksesCatalog.
- *+ getBukuParamISBN (catalog: AksesCatalog , isbn : string) : DataTable*
  Fungsi yang mereturnkan fungsi *getBukuParamISBN (isbn : string) : DataTable* pada AksesCatalog.
- *+ UpdateIdKategoriBuku (catalog: AksesCatalog , isbn : string, kategori: string) : bool*
  Fungsi yang mereturnkan fungsi *+ UpdateIdKategoriBuku (isbn : string, kategori: string) : bool* pada AksesCatalog.
- *+ ListBuku (catalog: AksesCatalog , pageNumber : string, out jmlTotalPage : int)*
- Fungsi yang mereturnkan fungsi *ListBuku (pageNumber : string, out jmlTotalPage : int)* pada AksesCatalog.

- *+ ListBukuParamIdKategori (catalog: AksesCatalog , idKategori:string, pageNumber:string, out jmlTotalPage:int)*
- Fungsi yang mereturnkan fungsi *ListBukuParamIdKategori (idKategori:string, pageNumber:string, out jmlTotalPage:int)* pada AksesCatalog.
- *+ ListBukuBaru (catalog: AksesCatalog , pageNumber : string)*
  Fungsi yang mereturnkan fungsi *ListBukuBaru (pageNumber : string)* pada AksesCatalog.
- *+ ListBukuBaruKlaster (catalog: AksesCatalog , pageNumber : string, username: string)*
  Fungsi yang mereturnkan fungsi *ListBukuBaruKlaster (pageNumber : string, username: string)* pada AksesCatalog.
- *+ ListSatuBukuBaru (catalog: AksesCatalog , pageNumber : string)*
  Fungsi yang mereturnkan fungsi *ListSatuBukuBaru (pageNumber : string)* pada AksesCatalog.
- *+ ListSatuBukuBaruKlaster (catalog: AksesCatalog , pageNumber : string, username: string)*
  Fungsi yang mereturnkan fungsi *ListSatuBukuBaruKlaster (pageNumber : string, username: string)* pada AksesCatalog.
- *+ ListBukuTerlaris (catalog: AksesCatalog , pageNumber : string)*
  Fungsi yang mereturnkan fungsi *ListBukuTerlaris (pageNumber : string)* pada AksesCatalog.
- *+ ListBukuTerlarisKlaster (catalog: AksesCatalog , pageNumber : string, username: string)*
  Fungsi yang mereturnkan fungsi *ListBukuTerlarisKlaster (pageNumber : string, username: string)* pada AksesCatalog.
- *+ ListSatuBukuTerlaris (catalog: AksesCatalog , pageNumber : string)*
  Fungsi yang mereturnkan fungsi *ListSatuBukuTerlaris (pageNumber : string)* pada AksesCatalog.
- *+ ListSatuBukuTerlarisKlaster (catalog: AksesCatalog , pageNumber : string, username: string)*
  Fungsi yang mereturnkan fungsi *ListSatuBukuTerlarisKlaster (pageNumber : string, username: string)* pada AksesCatalog.
- *+ ListSatuBukuJarangDiBeli (catalog: AksesCatalog , pageNumber : string)*
  Fungsi yang mereturnkan fungsi *ListSatuBukuJarangDiBeli (pageNumber : string)* pada AksesCatalog.
- *+ jumTotPageBukuBaru (catalog: AksesCatalog ,) : int*
  Fungsi yang mereturnkan fungsi *jumTotPageBukuBaru () : int* pada AksesCatalog.
- *+ jumTotPageBukuBaruKlaster (catalog: AksesCatalog , username: string) : int*
  Fungsi yang mereturnkan fungsi *jumTotPageBukuBaruKlaster (username: string) : int* pada AksesCatalog.
- *+ jumTotPageSatuBukuBaru (catalog: AksesCatalog,) : int*
  Fungsi yang mereturnkan fungsi *jumTotPageSatuBukuBaru () : int* pada AksesCatalog.
- *+ jumTotPageSatuBukuBaruKlaster (catalog: AksesCatalog) : int*
  Fungsi yang mereturnkan fungsi *jumTotPageSatuBukuBaruKlaster () : int* pada AksesCatalog.
- *+ jumTotPageSatuBukuJarangDiBeli (catalog: AksesCatalog,) : int*

Fungsi yang mereturnkan fungsi *jumTotPageSatuBukuJarangDiBeli () : int* pada AksesCatalog.

- *+ getKategoriParamIdKategori (catalog: AksesCatalog , idKategori : string) : DetailKategori*
  Fungsi yang mereturnkan fungsi *getKategoriParamIdKategori (idKategori : string) : DetailKategori* pada AksesCatalog.
- *+ Search (catalog: AksesCatalog , pageNumber : string, out jmlTotalPage, key : string, text : string) : DataTable*
  Fungsi yang mereturnkan fungsi *Search (pageNumber : string, out jmlTotalPage, key : string, text : string) : DataTable* pada AksesCatalog.
- *+ ListBukuKlaster (catalog: AksesCatalog , pageNumber : string, out jmlTotalPage : int, username: string)*
  Fungsi yang mereturnkan fungsi *ListBukuKlaster (pageNumber : string, out jmlTotalPage : int, username: string)* pada AksesCatalog.

2.2.12 Class MERISKA.AksesData.AksesOrderManager

<< Control >> AksesOrderManager
+StatusOrder: string[]
+GetOrderDetails (order: AksesOrder, idOrder: string): DataTable +getTotalHargaOrder (order: AksesOrder, idOrder: string): decimal +getStatusOrder (order: AksesOrder, idOrder: string): int +UpdateTglPengiriman(order: AksesOrder, idOrder: string) +OrderCancel(order: AksesOrder, idOrder: string) +getOrder(order: AksesOrder, IdOrder: string): AksesOrderInfo +ListGetOrderDetails (order: AksesOrder, orderId: string): List<AksesOrderDetailInfo> +CreateOrderPipeline (order: AksesOrder, IdOrder: int, pesan: string) +UpdateOrderStatus(order: AksesOrder, IdOrder: int, status: int) +SetTanggalPengiriman(order: AksesOrder, IdOrder: int) +GetOrderPipeline(order: AksesOrder, IdOrder: int): List<IdOrder: string> +GetOrdersByCustomer (order: AksesOrder, customerID: string): List<AksesOrderInfo> +GetOrdersByDate (order: AksesOrder, startDate: string, endDate: string): List<AksesOrderInfo> +UpdateOrder(order: AksesOrder, orderID: int, newTglOrder: string, newTglKirim: string, newStatus: int)

Gambar 2.11 Class MERISKA.AkssData.AksesOrderManager

Deskripsi

- Class yang berperan sebagai control class untuk mengimplementasikan fungsi-fungsi pada class AksesOrder, AksesOrderInfo, OrderPipeline dan AksesOrderDetailInfo ke userinterface:

Atribut

- *+ StatusOrder : string*
  Merepresentasikan status order.

Method

- *+ GetOrderDetail (order: AksesOrder, idOrder: string) : DataTable*
  Fungsi yang mereturnkan fungsi *GetOrderDetail (idOrder: string) : DataTable* pada AksesOrder.
- *+ ListGetOrderDetaisl (order: AksesOrder, orderId: string) : List<AksesOrderDetailInfo>*

Fungsi yang mereturnkan fungsi *ListGetOrderDetaisl (orderId: string) : List<AksesOrderDetailInfo>* pada AksesOrder.

- *+ GetTotalHargaOrder (order: AksesOrder, idOrder: string) : decimal*
  Fungsi yang mereturnkan fungsi *GetTotalHargaOrder (idOrder: string) : decimal* pada AksesOrder.
- *+ GetStatusOrder (order: AksesOrder, idOrder: string) : int*
  Fungsi yang mereturnkan fungsi *GetStatusOrder (idOrder: string) : int* pada AksesOrder.
- *+ UpdateTglPengiriman (order: AksesOrder, idOrder: string)*
  Fungsi yang mereturnkan fungsi *UpdateTglPengiriman (idOrder: string)* pada AksesOrder.
- *+ OrderCancel (order: AksesOrder, idOrder: string)*
  Fungsi yang mereturnkan fungsi *OrderCancel (idOrder: string)* pada AksesOrder.
- *+ GetOrder (order: AksesOrder, IdOrder: string) : AksesOrderInfo*
  Fungsi yang mereturnkan fungsi *GetOrder (IdOrder: string) : AksesOrderInfo.*
- *+ UpdateOrderStatus (order: AksesOrder, IdOrder: int, status: int)*
  Fungsi yang mereturnkan fungsi *UpdateOrderStatus (IdOrder: int, status: int)* pada AksesOrder.
- *+ SetTglPengiriman (order: AksesOrder, idOrder: int)*
  Fungsi yang mereturnkan fungsi *SetTglPengiriman (idOrder: int).*
- *+ GetOrderPipeline (order: AksesOrder, IdOrder: string) : List<OrderPipeline>*
  Fungsi yang mereturnkan fungsi *GetOrderPipeline (IdOrder: string) : List<OrderPipeline>* pada AksesOrder.
- *+ GetOrderByCustomer (order: AksesOrder, customerID: string) : List<AksesOrderInfo>*
  Fungsi yang mereturnkan fungsi *GetOrderByCustomer (customerID: string) : List<AksesOrderInfo>* pada AksesOrder.
- *+ GetOrderByDate (order: AksesOrder, startDate: string, endDate: string) : List<AksesOrderInfo>*
  Fungsi yang mereturnkan fungsi *GetOrderByDate (startDate: string, endDate: string) : List<AksesOrderInfo>* pada AksesOrder.
- *+ UpdateOrder (order: AksesOrder, orderID: int, newTglOrder: string, newTglKirim: string, newStatus: int)*
  Fungsi yang mereturnkan fungsi *UpdateOrder (orderID: int, newTglOrder: string, newTglKirim: string, newStatus: int)* pada AksesOrder.
- *+ CreateOrderPipeline (order: AksesOrder, IdOrder: int, pesan: string)*
  Fungsi yang mereturnkan fungsi *CreateOrderPipeline (IdOrder: int, pesan: string)* pada AksesOrder.

2.2.13 Class MERISKA.AksesData.AksesKeranjangBelanjaManager

<< Control>> AksesKeranjangBelanjaManager
+AddKeranjangBelanja (keranjang: AksesKeranjangBelanja, isbn: string): bool +UpdateKeranjangBelanja (keranjang: AksesKeranjangBelanja, isbn: string, quantity: int): bool +DeleteKeranjangBelanja (keranjang: AksesKeranjangBelanja, isbn: string): bool +getKeranjangBelanja (keranjang: AksesKeranjangBelanja): DataTable

+getTotalHarga (keranjang: AksesKeranjangBelanja): decimal +getKeranjangBelanjaAdmin (keranjang: AksesKeranjangBelanja, hari: byte): DataTable +getDetailKeranjangBelanjaAdmin (keranjang: AksesKeranjangBelanja, Id: string): DataTable +DeleteKeranjangBelanjaAdmin (keranjang: AksesKeranjangBelanja, Id: string): DataTable +getQuantity (keranjang: AksesKeranjangBelanja, isbn: string): int +CreateOrder (keranjang: AksesKeranjangBelanja): string

Gambar 2.12 Class MERISKA.AksesData.AksesKeranjangBelanjaManager

Deskripsi

Class yang berperan sebagai control class untuk mengimplementasikan fungsi-fungsi pada class AksesKeranjangBelanja ke userinterface:

Atribut

-

Method

- *+ AddKeranjangBelanja (keranjang: AksesKeranjangBelanja, isbn : string) : bool*
  Fungsi yang mereturnkan fungsi *AddKeranjangBelanja (isbn : string) : bool* pada AksesKeranjangBelanja.
- *+ UpdateKeranjangBelanja (keranjang: AksesKeranjangBelanja, isbn : string, quantity : int) : bool*
  Fungsi yang mereturnkan fungsi *UpdateKeranjangBelanja(isbn : string, quantity : int) : bool* pada AksesKeranjangBelanja.
- *+ DeleteKeranjangBelanja (keranjang: AksesKeranjangBelanja, isbn: string) : bool*
  Fungsi yang mereturnkan fungsi *DeleteKeranjangBelanja (isbn: string) : bool* pada AksesKeranjangBelanja.
- *+ getKeranjangBelanja(keranjang: AksesKeranjangBelanja) : DataTable*
  Fungsi yang mereturnkan fungsi *getKeranjangBelanja () : DataTable* pada AksesKeranjangBelanja.
- *+ getTotalHarga (keranjang: AksesKeranjangBelanja) : DataTable*
  Fungsi yang mereturnkan fungsi *getTotalHarga () : DataTable* pada AksesKeranjangBelanja.
- *+ getKeranjangBelanjaAdmin (keranjang: AksesKeranjangBelanja, hari: byte) : DataTable*
  Fungsi yang mereturnkan fungsi *getKeranjangBelanjaAdmin (hari: byte) : DataTable* pada AksesKeranjangBelanja.
- *+ getDetailKeranjangBelanjaAdmin (keranjang: AksesKeranjangBelanja, Id: string) : DataTable*
  Fungsi yang mereturnkan fungsi *getDetailKeranjangBelanjaAdmin (Id: string) : DataTable* pada AksesKeranjangBelanja.
- *+ DeleteKeranjangBelanjaAdmin (keranjang: AksesKeranjangBelanja, Id: string) : bool*
  Fungsi yang mereturnkan fungsi *DeleteKeranjangBelanjaAdmin (: string) : bool* pada AksesKeranjangBelanja.
- *+ getQuantity (keranjang: AksesKeranjangBelanja, isbn: string) : int*
  Fungsi yang mereturnkan fungsi *getQuantity (isbn: string) : int* pada AksesKeranjangBelanja.
- *+ CreateOrder (keranjang: AksesKeranjangBelanja) : string*
  Fungsi yang mereturnkan fungsi *CreateOrder () : string* pada AksesKeranjangBelanja.

2.2.14 Class MERISKA.AksesData.AksesKlasterManager

<< Control >> AksesKlastermanager
+getKlaster (klaster: AksesKlaster): DataTable +UpdateKlaster (klaster: AksesKlaster, centroid: string, keterangan: string): bool

Gambar 2.13 Class MERISKA.AksesData.AksesKlasterManager

Deskripsi

Class yang berperan sebagai control class untuk mengimplementasikan fungsi-fungsi pada class AksesKlaster ke userinterface:

Atribut

-

Method

- *+ getKlaster (klaster: AksesKlaster) : DataTable*
  Fungsi yang mereturnkan fungsi *getKlaster () : DataTable* pada AksesKlaster.
- *+ UpdateKlaster (klaster: AksesKlaster, centroid: string, keterangan: string)*
  Fungsi yang mereturnkan fungsi *UpdateKlaster (centroid: string, keterangan: string)* pada AksesKlaster.

2.2.15 Class MERISKA.AksesData.AksesStokManager

<< Entity >> AksesStokManager
+AddStok (stok: AksesStok, Isbn: string): bool +UpdateStok (stok: AksesStok, Isbn: string, Quantity: string): bool +getStokBuku (stok: AksesStok, isbn: string): int +getDataStok (stok: AksesStok): DataTable +getStokParamJudul (stok: AksesStok, judul: string): DataTable +stokEmpty (stok: AksesStok, isbn: string): bool

Gambar 2.14 Class MERISKA.AksesData.AksesKlasterManager

Deskripsi

Class yang berperan sebagai control class untuk mengimplementasikan fungsi-fungsi pada class AksesStok ke userinterface:

Atribut

-

Method

- *+ AddStok (stok: AksesStok, Isbn : string) : bool*
  Fungsi yang mereturnkan fungsi *AddStok (Isbn : string) : bool* pada AksesStok.
- *+ UpdateKategori (stok: AksesStok, Isbn: string, Quantity : string) : bool*
  Fungsi yang mereturnkan fungsi *UpdateKategori (Isbn: string, Quantity : string) : bool* pada AksesStok.
- *+ getStokBuku (stok: AksesStok, Isbn : string) : int*
  Fungsi yang mereturnkan fungsi *getStokBuku (Isbn : string) : int* pada AksesStok.
- *+ getDataStok (stok: AksesStok) : DataTable*
  Fungsi yang mereturnkan fungsi *getDataStok () : DataTable* pada AksesStok.
- *+ getStokParamJudul (stok: AksesStok , judul: string) : DataTable*

Fungsi yang mereturnkan fungsi *getStokParamJudul (judul: string) : DataTable* pada AksesStok.

- *+ stokEmpty (stok: AksesStok , Isbn : string) : bool*
  Fungsi yang mereturnkan fungsi *stokEmpty (Isbn : string) : bool* pada AksesStok.

2.2.16 Class MERISKA.Profil.ProfilData

<< Entity >> ProfilData
-nama : string -alamat : string -noHp : string -noTelpRumah : string -email : string +Nama : string +Alamat : string +NoHp : string +NoTelpRumah : string +Email : string
+ProfilData() +UpdateProfil ()

Gambar 2.15 Class MERISKA.Profil.ProfilData

Deskripsi

Class yang berperan sebagai entity class yang merepresentasikan profil data customer.

Atribut

- *- nama : string*
  Merepresentasikan nama customer dan bersifat private.
- *- alamat : string*
  Merepresentasikan alamat customer dan bersifat private.
- *- noHp : string*
  Merepresentasikan no hp customer dan bersifat private.
- *- noTelpRumah : string*
  Merepresentasikan no telp rumah customer dan bersifat private.
- *- email : string*

Method

- *+ ProfilData ()*
  Merupakan konstruktor dari kelas profil data dan berisi value dari atribut-atribut yang bersifat private.
  Merepresentasikan email customer dan bersifat private.
- *+ Nama : string*
  Method untuk set dan get nama customer dan bersifat public.
- *+ Alamat : string*
  Method untuk set dan get alamat customer dan bersifat public.
- *+ NoHp : string*
  Method untuk set dan get no hp customer dan bersifat public.
- *+ NoTelpRumah : string*
  Method untuk set dan get no telp rumah customer dan bersifat public.
- *+ Email : string*

Method untuk set dan get email customer dan bersifat public

- *+ UpdateProfil ()*
  Prosedur yang berfungsi mengubah profil customer.
  Parameters:
  -
  Return:
  -

2.2.17 Class MERISKA.Profil.ProfilManager

<< Control >> ProfilManager
+GetData(): List<ProfilData> +UpdateData (newData: ProfilData)

Gambar 2.16 Class MERISKA.Profil.ProfilManager

Deskripsi

Class yang berperan sebagai control class untuk aksi yang berhubungan dengan akses stok ke basis data. Aksi yang berhubungan dengan class ini yaitu:

- Get data.
- Update data

Atribut

-

Method

- *+ GetData () : List<ProfilData>*
  Digunakan untuk mengambil list data profil.
  Parameters:
  -
  Return:
  List data profil.
- *+ UpdateData (newData: ProfilData)*
  Mengubah list data profil.
  Parameters:
  newData – merupakan parameter inputan untuk attribute data profil yang akan diubah.
  Return:
  -

2.2.18 Class MERISKA.NotifikasiOrderPembelian.OrderProcessor

<< Control >> OrderProcessor
+CurrentPipeline: iOrderPipeline +Continue: bool +order: AksesOrdeInfo +entitas_order: AksesOrder
+OrderProcessor(IdOrder: string) +OrderProcessor(order: AksesOrderInfo) +Process() +CreateOrderPipeline(pesan: string)

+MailAdmin(subject: string, pesan: string) +MailCustomer(subject: string, pesan: string) +GetCurrentPipeline()

Gambar 2.17 Class MERISKA.NotifikasiOrderPembelian.OrderProcessor

Deskripsi

Class yang berperan sebagai control class. Aksi yang berhubungan dengan kelas ini yaitu:

- Proses order
- Pipeline

Atribut

- *- CurrentPipeline : iOrderPipeline*
  Merepresentasikan pipeline sekarang.
- *- Continue: bool*
  Merepresentasikan proses continue pipeline.
- *- order : AksesOrderInfo*
  Merepresentasikan Info order.
- *- entitas_order : AksesOrder*
  Merepresentasikan order.

Method

- *+ OrderProcessor (IdOrder: string)*
  Merupakan konstruktor dengan parameter Id order.
- *+ OrderProcessor (order: AksesOrderInfo)*
  Merupakan konstruktor dengan parameter order.
- *+ Process ()*
  Merupakan fungsi untuk proses order.
- *+ CreateOrderPipeline (pesan: string)*
  Merupakan fungsi untuk proses pipeline.
- *+ MailAdmin (subject: string, pesan: string)*
  Merupakan fungsi untuk mengirim pesan ke admin melalui proses order.
- *+ MailCustomer (subject: string, pesan: string)*
  Merupakan fungsi untuk mengirim pesan ke customer melalui proses order.
- *+ GetCurrentPipeline ()*
  Merupakan fungsi untuk mendapatkan current pipeline.

2.2.19 Class MERISKA.Kontrol.BukuAdmin

<< Boundary >> BukuAdmin
+BindGrid ()

Gambar 2.18 Class MERISKA.Kontrol.BukuAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan pengelolahan buku admin.

Atribut

-

Method

- *+ BindGrid ()*
  Fungsi untuk mengisi data grid dengan list buku.

Parameters:

-

Return:

-

2.2.20 Class MERISKA.Kontrol.DetailBukuAdmin

<< Boundary >> DetailBukuAdmin
+ isbn: string
+IsiData ()

Gambar 2.19 Class MERISKA.Kontrol.DetailBukuAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan pengelolahan detail buku admin.

Atribut

-

Method

- *+ BindGrid ()*

  Fungsi untuk mengisi data grid dengan list buku.

  Parameters:

  -

  Return:

  -

2.2.21 Class MERISKA.Kontrol.DetailKeranjangBelanjaAdmin

<< Boundary >> DetailKeranjangBelanjaAdmin
+BindGrid ()

Gambar 2.20 Class MERISKA.Kontrol.DetailKeranjangBelanjaAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan detail keranjang belanja admin.

Atribut

-

Method

- *+ BindGrid ()*

  Fungsi untuk mengisi data grid dengan list detail keranjang belanja.

2.2.22 Class MERISKA.Kontrol.DetailOrderAdmin

<< Boundary >> DetailOrderAdmin
-editMode: bool
+IsiData () +SetEditMode (enable: bool)

Gambar 2.21 Class MERISKA.Kontrol.DetailOrderAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan pengelolahan detail order admin.

Atribut

- *- editMode: bool*

  Merupakan attribute untuk mengeset fungsi edit, defaultnya false.

Method

- *+ IsiData ()*

  Fungsi untuk mengisi data pada setiap kontrol UI (pada textboxt, dropdown list dan datagrid) .
- *+ SetEditMode (enable: bool)*

  Fungsi untuk mengeset edit mode .

2.2.23 Class MERISKA.Kontrol.EditKlaster

<< Boundary >> EditKlaster
+BindGrid ()

Gambar 2.22 Class MERISKA.Kontrol.EditKlaster

Deskripsi

Class ini merupakan UI untuk menampilkan fungsi edit label klaster.

Atribut

-

Method

- *+ BindGrid ()*

  Fungsi untuk mengisi data grid dengan list klaster.

2.2.24 Class MERISKA.Kontrol.KategoriAdmin

<< Boundary >> KategoriAdmin
+BindGrid ()

Gambar 2.23 Class MERISKA.Kontrol.KategoriAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan pengelolahan kategori admin.

Atribut

-

Method

- *+ BindGrid ()*

  Fungsi untuk mengisi data grid dengan list kategori.

2.2.25 Class MERISKA.Kontrol.KeranjangBelanjaAdmin

<< Boundary >> KeranjangBelanjaAdmin
+ hari: byte
+BindGrid (hari: byte)

Gambar 2.24 Class MERISKA.Kontrol.KeranjangBelanjaAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan pengelolahan keranjang belanja admin.

Atribut

- *- hari: byte*

  Sebagai attribute untuk membatasi jumlah list keranjang belanja yang akan ditampilkan(berdasarkan lama hari).

Method

- *+ BindGrid (hari: byte)*

  Fungsi untuk mengisi data grid dengan list keranjang belanja berdasarkan lama hari.

2.2.26 Class MERISKA.Kontrol.KeranjangDetail

<< Boundary >> KeranjangDetail
+IsiData()

Gambar 2.25 Class MERISKA.Kontrol.KeranjangDetail

Deskripsi

Class ini merupakan UI untuk menampilkan detail keranjang belanja.

Atribut

-

Method

- *+ IsiData ()*

  Fungsi untuk mengisi data pada setiap kontrol UI (pada textboxt dan datalist)

2.2.27 Class MERISKA.Kontrol.ListBuku

<< Boundary >> ListBuku
+IsiData()

Gambar 2.26 Class MERISKA.Kontrol.ListBuku

Deskripsi

Class ini merupakan UI untuk menampilkan list buku.

Atribut

-

Method

- *+ IsiData ()*

  Fungsi untuk mengisi data buku pada kontrol UI (datalist)

2.2.28 Class MERISKA.Kontrol.ListBukuBaru

<< Boundary >> ListBukuBaru
+IsiData()

Gambar 2.27 Class MERISKA.Kontrol.ListBukuBaru

Deskripsi

Class ini merupakan UI untuk menampilkan list buku baru.

Atribut

-

Method

- + *IsiData ()*

  Fungsi untuk mengisi data buku baru pada kontrol UI (datalist)

2.2.29 Class MERISKA.Kontrol.ListBukuTerlaris

<< Boundary >> ListBukuTerlaris
+IsiData()

Gambar 2.28 Class MERISKA.Kontrol.ListBukuTerlaris

Deskripsi

Class ini merupakan UI untuk menampilkan list buku terlaris.

Atribut

-

Method

- + *IsiData ()*

  Fungsi untuk mengisi data buku terlaris pada kontrol UI (datalist)

2.2.30 Class MERISKA.Kontrol.Listkategori

<< Boundary >> ListKategori
+BindGrid()

Gambar 2.29 Class MERISKA.Kontrol.ListKategori

Deskripsi

Class ini merupakan UI untuk menampilkan list kategori.

Atribut

-

Method

- + BindGrid ()

  Fungsi untuk mengisi data grid dengan list kategori.

2.2.31 Class MERISKA.Kontrol.ListSatuBukuBaru

<< Boundary >> ListSatuBukuBaru
+IsiData()

Gambar 2.30 Class MERISKA.Kontrol.ListSatuBukuBaru

Deskripsi

Class ini merupakan UI untuk menampilkan list satu buku baru.

Atribut

-

Method

- + *IsiData ()*

Fungsi untuk mengisi data satu buku baru pada kontrol UI (datalist)

2.2.32 Class MERISKA.Kontrol.ListSatuBukuJarangDiBeli

<< Boundary >> ListSatuBukuJarangDiBeli
+IsiData()

Gambar 2.31 Class MERISKA.Kontrol.ListSatuBukuJarangDiBeli

Deskripsi

Class ini merupakan UI untuk menampilkan satu buku jarang dibeli.

Atribut

-

Method

- *+ IsiData ()*
  Fungsi untuk mengisi data satu buku jarang di beli pada kontrol UI (datalist)

2.2.33 Class MERISKA.Kontrol.ListSatuBukuTerlaris

<< Boundary >> ListSatuBukuTerlaris
+IsiData()

Gambar 2.32 Class MERISKA.Kontrol.ListSatuBukuTerlaris

Deskripsi

Class ini merupakan UI untuk menampilkan list satu buku terlaris.

Atribut

-

Method

- *+ IsiData ()*
  Fungsi untuk mengisi data satu buku terlaris pada kontrol UI (datalist)

2.2.34 Class MERISKA.Kontrol.MailAdmin

<< Boundary >> MailAdmin
+BindGrid() +harga() +refresh()

Gambar 2.33 Class MERISKA.Kontrol.MailAdmin

Deskripsi

Class ini merupakan UI untuk menampilkan pengelolahan mail admin.

Atribut

-

Method

- *+ BindGrid ()*
  Fungsi untuk mengisi datagrid dengan list detail order.
- *+ harga ()*
  Fungsi yang mereturnkan total harga order.

- *+refresh ()*
  Fungsi untuk mengisi data pada kontrol UI (pada textbox, dropdown list, label).

2.2.35 Class MERISKA.Kontrol.Search

<< Boundary >> Search
+Search()

Gambar 2.34 Class MERISKA.Kontrol.Search

Deskripsi

Class ini merupakan UI untuk menampilkan pencarian data.

Atribut

-

Method

- *+Search ()*
  Fungsi untuk redirect ke page search berdasarkan pencarian yang diinputkan.

2.2.36 Class Utilities (MERISKA.Utilities)

<< Control >> Utilities
+TieButton (page : Page, TextBoxToTie : Control, ButtonToTie : Control) +Random (randObj : Random, max : int) : string +SendMail (from: string, to: string, subject: string, body: string)

Gambar 2.35 Class MERISKA.Utilities

Deskripsi

Merupakan control class yang menampung fungsi-fungsi bantu untuk program utama.

Atribut

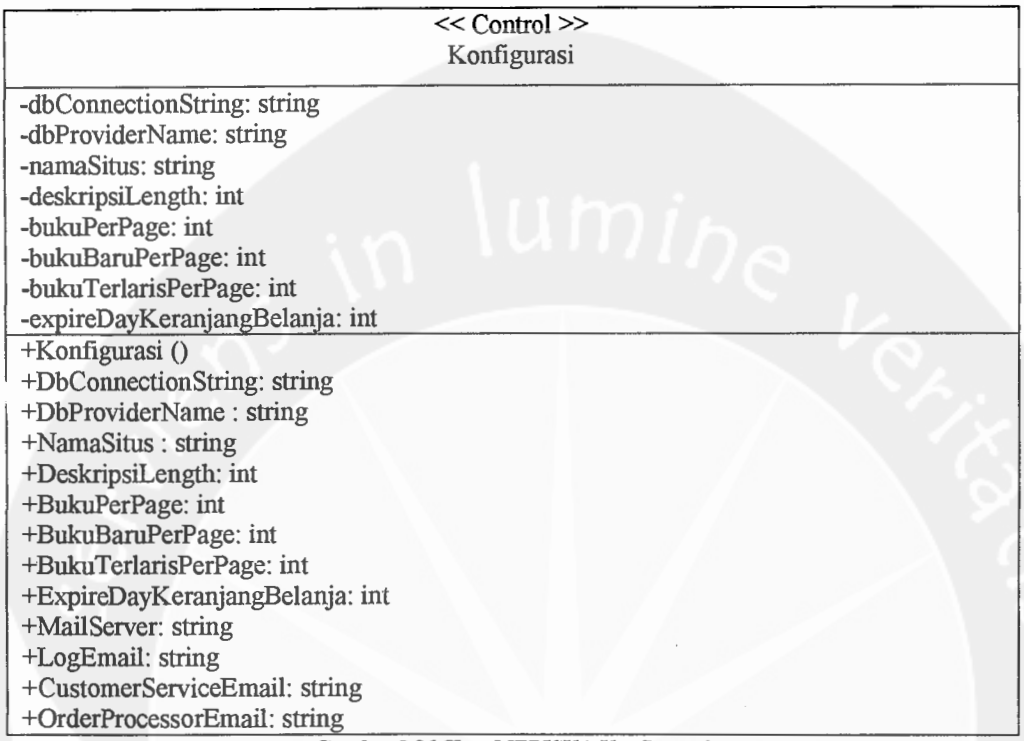
-

Method

- + TieButton (page : Page, TextBoxToTie : Control, ButtonToTie : Control)
  Merupakan fungsi untuk menyamakan event keypress pada keyboard dan click pada mouse.
  Parameters:
  TextBoxToTie dan ButtonToTie - keduanya merupakan parameter yang bertipe control, Page - merupakan parameter yang merepresentasikan page dimana event tersebut terjadi
- + Random (randObj : Random, max : int)
  Merupakan fungsi untuk melakukan rando terhadap objek.
  Parameters:
  randObj – objek yang akan di random.
  max – maximum nilai yang dirandom.
- + SendMail (from: string, to: string, subject: string, body: string)
  Merupakan fungsi untuk mengirim email.
  Parameters:

from – merupakan parameter untuk *from* pada email.
to – merupakan parameter untuk *to* pada email.
subject – merupakan parameter untuk *subject* pada email.
body – merupakan parameter untuk *body* pada email.

2.2.37 Class Konfigurasi (MERISKA.Konfigurasi)



Gambar 2.36 Class MERISKA.Konfigurasi

Deskripsi

Class yang berperan sebagai control class yang mengatur konfigurasi umum sistem.

Atribut

- *- dbConnectionString: string*
  Merepresentasikan database connection string dan bersifat private.
- *- dbProviderName: string*
  Merepresentasikan database provide name dan bersifat private.
- *- namaSitus: string*
  Merepresentasikan nama situs dan bersifat private.
- *- deskripsiLength: int*
  Merepresentasikan panjang deskripsi buku yang akan ditampilkan dan bersifat private.
- *- bukuPerPage: int*
  Merepresentasikan jumlah buku per page dan bersifat private.
- *- bukuBaruPerPage: int*
  Merepresentasikan jumlah buku baru per page dan bersifat private.
- *- bukuTerlarisPerPage: int*
  Merepresentasikan jumlah buku terlaris per page dan bersifat private.
- *- expireDayKeranjangBelanja: int*
  Merepresentasikan expire day keranjang belanja dan bersifat private.

Method

- *+ Konfigurasi ()*
  Merupakan konstruktor dari kelas konsfigurasi dan berisi value dari atribut-atribut yang bersifat private.
- *+DbConnectionString: string*
  Method untuk get database connection string dan bersifat public.
- *+DbProviderName: string*
  Method untuk get database provider name dan bersifat public.
- *+NamaSitus: string*
  Method untuk get nama situs dan bersifat public.
- *+DeskripsiLength: string*
  Method untuk get deskripsi length dan bersifat public.
- *+BukuPerPage: int*
  Method untuk get jumlah buku per page dan bersifat public.
- *+BukuBaruPerPage: int*
  Method untuk get jumlah buku baru per page dan bersifat public.
- *+BukuTerlarisPerPage: int*
  Method untuk get jumlah buku baru per page dan bersifat public.
- *+ExpireDayKeranjangBelanja : int*
  Method untuk get expire day keranjang belanja dan bersifat public.
- *+MailServer: string*
  Method untuk get mail server dan bersifat public.
- *+LogEmail: string*
  Method untuk get log email dan bersifat public.
- *+CustomerServiceEmail: string*
  Method untuk get customer service email dan bersifat public.
- *+OrderProcessorEmail: string*
  Method untuk get order processor email dan bersifat public.

2.2.38 Class iOrderPipeline (MERISKA.NotifikasiOrderPembelian.iOrderPipeline)

<< control >> iOrderPipeline
-
+iOrderPipeline : interface

Gambar 2.37 Class MERISKA.NotifikasiOrderPembelian.iOrderPipeline

Deskripsi

Class yang berperan sebagai interface class.

Atribut

-

Method

- *+ iOrderPipeline : interface*
  Method yang berisi fungsi abstrak.

2.2.39 Class NotifikasiAwalMail (MERISKA.NotifikasiOrderPembelian.NotifikasiAwalMail)

<< Control >> NotifikasiAwalMail

- orderProcessor: orderProcessor
+ Proses (processor: OrderProcessor) - GetMailBody() : string

Gambar 2.38 Class MERISKA.NotifikasiOrderPembelian.NotifikasiAwalMail

Deskripsi

Class yang berisi method proses notifikasi awal order dan fungsi get mail body

Atribut

- *- orderProcessor: orderProcessor*
  Merepresentasikan orderProcessor dan bersifat private.

Method

- *+ Proses (processor: OrderProcessor)*
  Method yang berisi proses notifikasi awal order.
- *+ GetMailBody () : string*
  Method untuk mendapatkan body email.

2.2.40 Class NotifikasiPembayaranTelahDiTerimaMail (MERISKA.NotifikasiOrderPembelian.NotifikasiPembayaranTelahDiTerimaMail)

<< Control >> NotifikasiPembayaranTelahDiTerimaMail
- orderProcessor: orderProcessor
+ Proses (processor: OrderProcessor) - GetMailBody() : string

Gambar 2.39 Class MERISKA.NotifikasiOrderPembelian.NotifikasiPembayaranTelahDiTerimaMail

Deskripsi

Class yang berisi method proses notifikasi pembayaran telah diterima dan fungsi get mail body

Atribut

- *- orderProcessor: orderProcessor*
  Merepresentasikan orderProcessor dan bersifat private.

Method

- *+ Proses (processor: OrderProcessor)*
  Method yang berisi proses notifikasi pembayaran telah diterima.
- *+ GetMailBody () : string*
  Method untuk mendapatkan body email.

2.2.41 Class NotifikasiPengirimanMail (MERISKA.NotifikasiOrderPembelian.NotifikasiPengirimanMail)

<< Control >> NotifikasiPengirimanMail
- orderProcessor: orderProcessor
+ Proses (processor: OrderProcessor) - GetMailBody() : string

Gambar 2.40 Class MERISKA.NotifikasiOrderPembelian.NotifikasiPengirimanMail

Deskripsi

Class yang berisi method proses notifikasi pengiriman dan fungsi get mail body

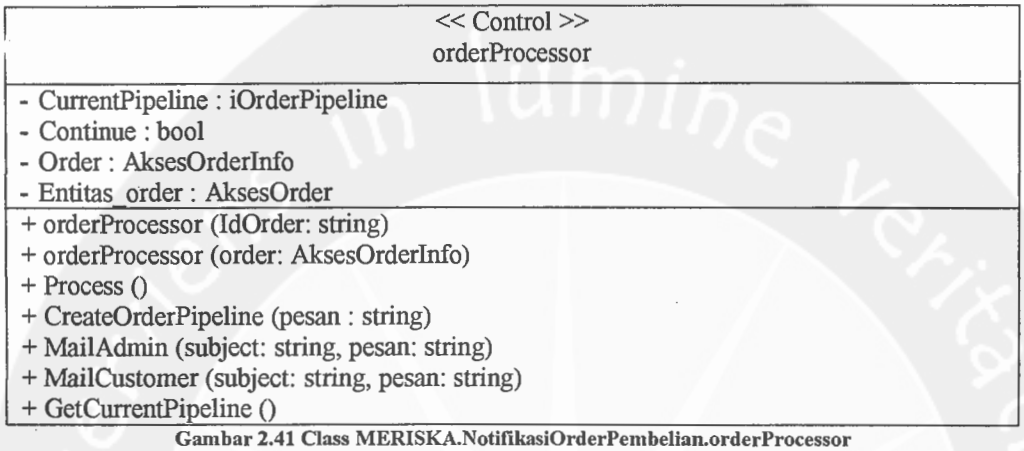
Atribut

- *- orderProcessor: orderProcessor*
  Merepresentasikan orderProcessor dan bersifat private.

Method

- *+ Proses (processor: OrderProcessor)*
  Method yang berisi proses notifikasi pengiriman.
- *+ GetMailBody () : string*
  Method untuk mendapatkan body email.

2.2.42 Class orderProcessor (MERISKA.NotifikasiOrderPembelian.orderProcessor)



Gambar 2.41 Class MERISKA.NotifikasiOrderPembelian.orderProcessor

Deskripsi

Class yang menangani fungsi-fungsi yang berhubungan dengan proses order

Atribut

- *- CurrentPipeline: iOrderPipeline*
  Merepresentasikan interface iOrderPipeline dan bersifat private.
- *- Continue: bool*
  Merupakan attribute untuk mengeset fungsi continue proses order.
- *- Order: AksesOrderInfo*
  Merepresentasikan kelas AksesOrderInfo dan bersifat private.
- *- entitas_order: AkseOrder*
  Merepresentasikan kelas AksesOrder dan bersifat private.

Method

- *+ orderProcessor (IdOrder: string)*
  Merupakan kelas konstruktor dengan parameter Id order.
- *+ orderProcessor (order: AksesOrderInfo)*
  Merupakan kelas konstruktor dengan parameter order.
- *+ Process ()*
  Merupakan method untuk proses order.
- *+ CreateOrderPipeline (pesan: string)*
  Merupakan method untuk mencatat track/log proses order.
- *+ MailAdmin (subject: string, pesan: string)*
  Merupakan method untuk mengirim emai ke admin.
- *+ MailCustomer (subject: string, pesan: string)*
  Merupakan method untuk mengirim emai ke customer.
- *+ GetCurrentPipeline ()*

Merupakan method untuk mendapatkan status proses order yang sedang terjadi.

2.2.43 Class OrderProcessorMailer (MERISKA.NotifikasiOrderPembelian.orderProcessorMailer)

<< Control >> OrderProcessorMailer
-
+ MailAdmin (IdOrder: int, subject: string, pesan: string) + MailCustomer (customer: MembershipUser, subject: string, pesan: string) + MailService (to: string, subject: string, pesan: string)

Gambar 2.42 Class MERISKA.NotifikasiOrderPembelian.OrderProcessorMailer

Deskripsi

Class yang menangani fungsi-fungsi yang berhubungan dengan email.

Atribut

-

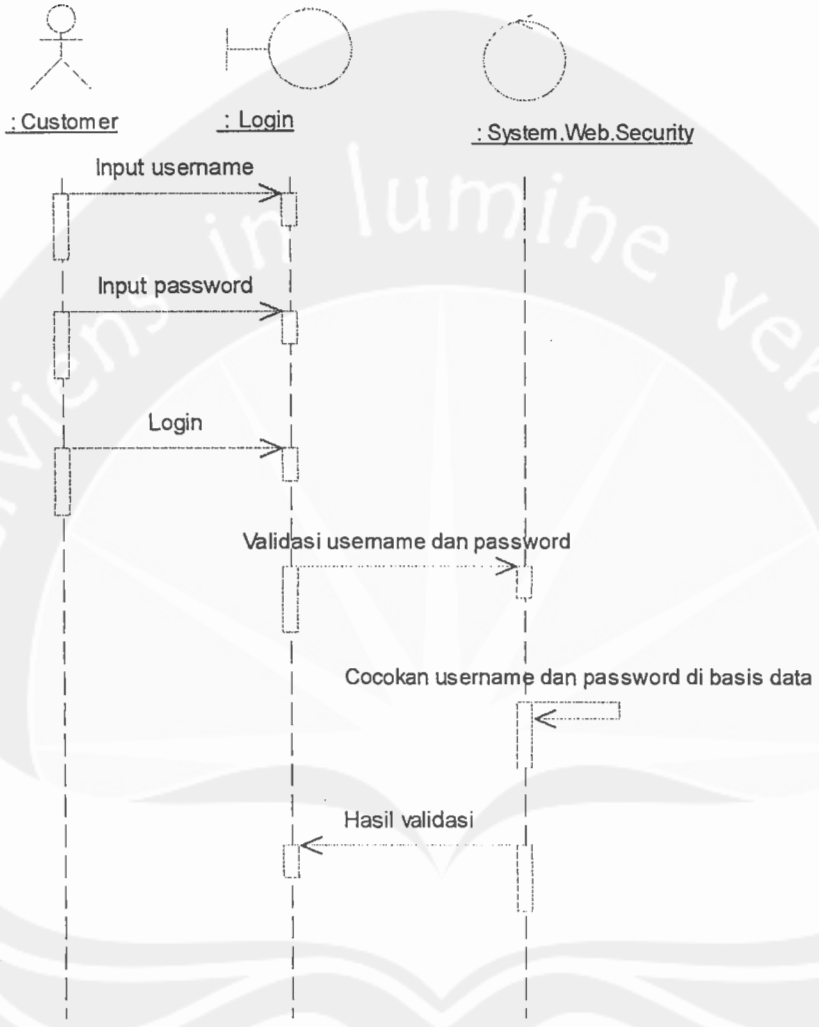
Method

- *+ MailAdmin (IdOrder: int, subject: string, pesan: string)*
  Merupakan method untuk setting email yang akan dikirim ke admin.
- *+ MailCustomer (customer: MembershipUser, subject: string, pesan: string)*
  Merupakan method untuk setting email yang akan dikirim ke customer.
- *+ MailService (to: stringr, subject: string, pesan: string)*
  Merupakan method untuk setting email yang akan dikirim ke customer service.

2.3 Realisasi Use Case

2.3.1 Use Case : Login

2.3.1.1 Login

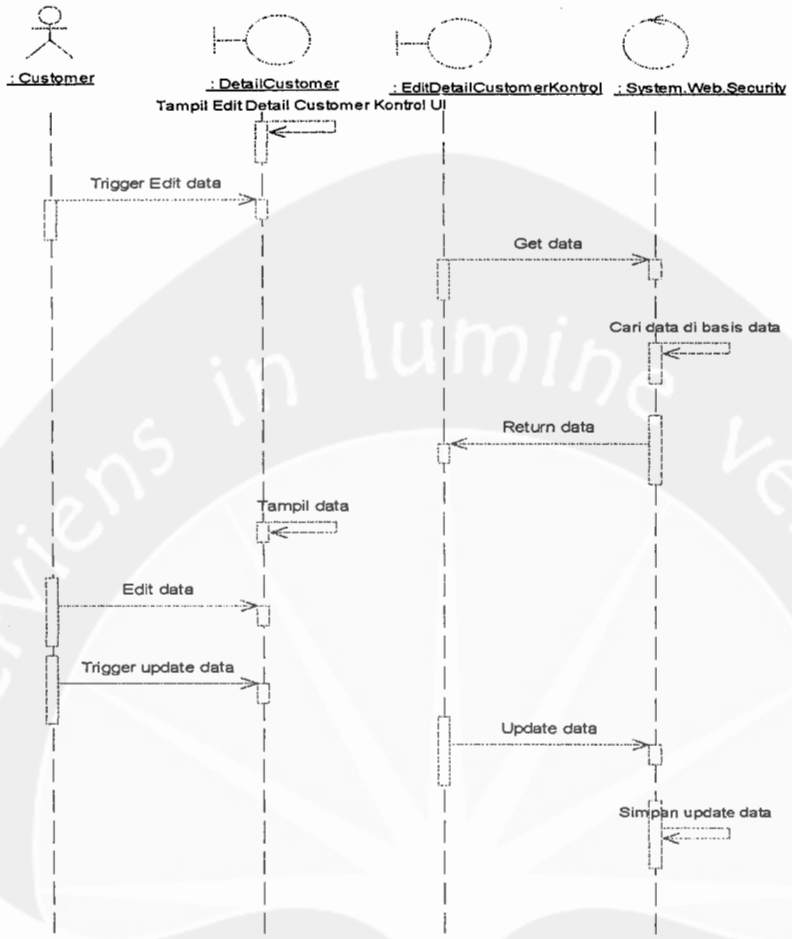


Gambar 2.43 Design Sequence Diagram : Use Case Login

Flow of events :

1. Customer menginputkan username dan password pada antarmuka login `loginUI` yang ditampilkan kemudian melakukan aksi login.
2. `loginUI` akan melakukan validasi username dan password dengan memanggil fungsi yang terdapat pada `System.Web.Security`.
3. `System.Web.Security` akan mencocokan inputan dengan data yang terdapat pada basis data.
4. `System.Web.Security` memberikan hasil validasi ke customer melalui loginUI.

2.3.2 Use Case : Pengubahan Profil Customer

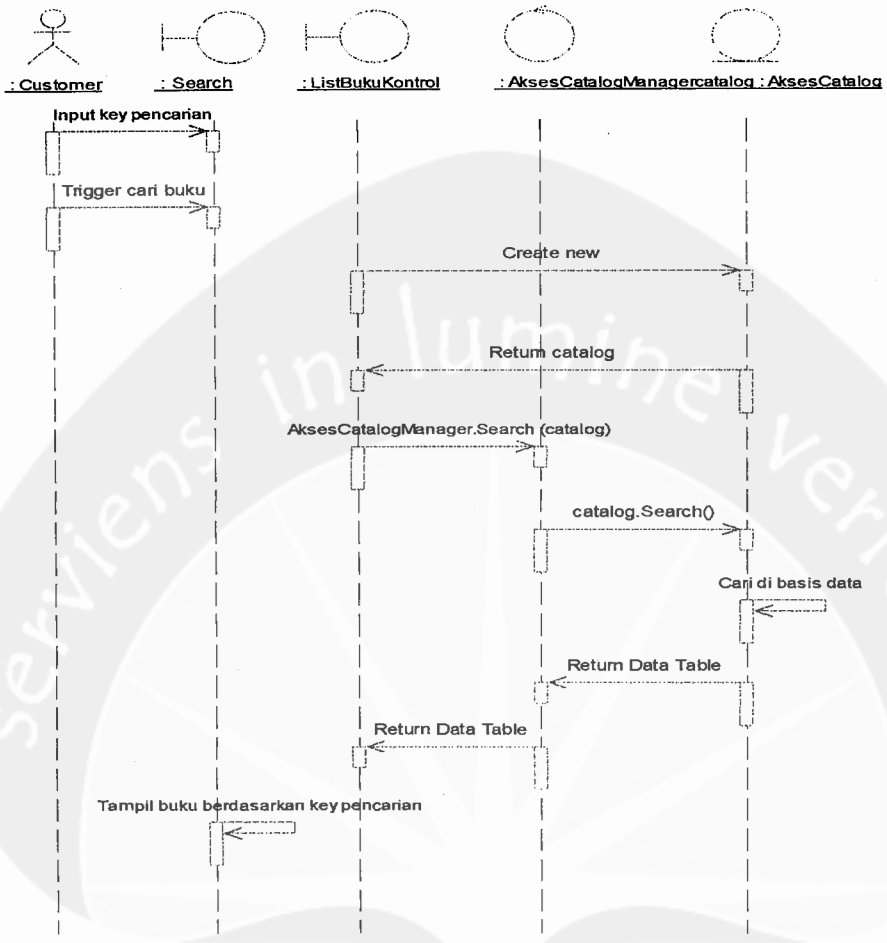


Gambar 2.44 Design Sequence Diagram : Use Case Pengubahan Profil Customer

Flow of events :

1. `detailCustomerUI` menampilkan kontrol `EditDetailCustomerUI` untuk edit customer profil.
2. Customer mentrigger edit data pada antarmuka `EditDetailCustomerUI` yang ditampilkan.
3. `EditDetailCustomerUI` akan mengambil *current data* customer dari basis data dengan memanggil fungsi list *get data* dari `System.Web.Security`
4. `System.Web.Security` akan mereturnkan data ke UI.
5. UI menampilkan data ke customer.
6. Customer akan melakukan proses ubah data kemudian melakukan mentrigger update data.
7. `EditDetailCustomerUI` memanggil fungsi *update data* dari `System.Web.Security`.
8. `System.Web.Security` menyimpan profil yang telah diedit.

2.3.3 Use Case : Pencarian Catalog



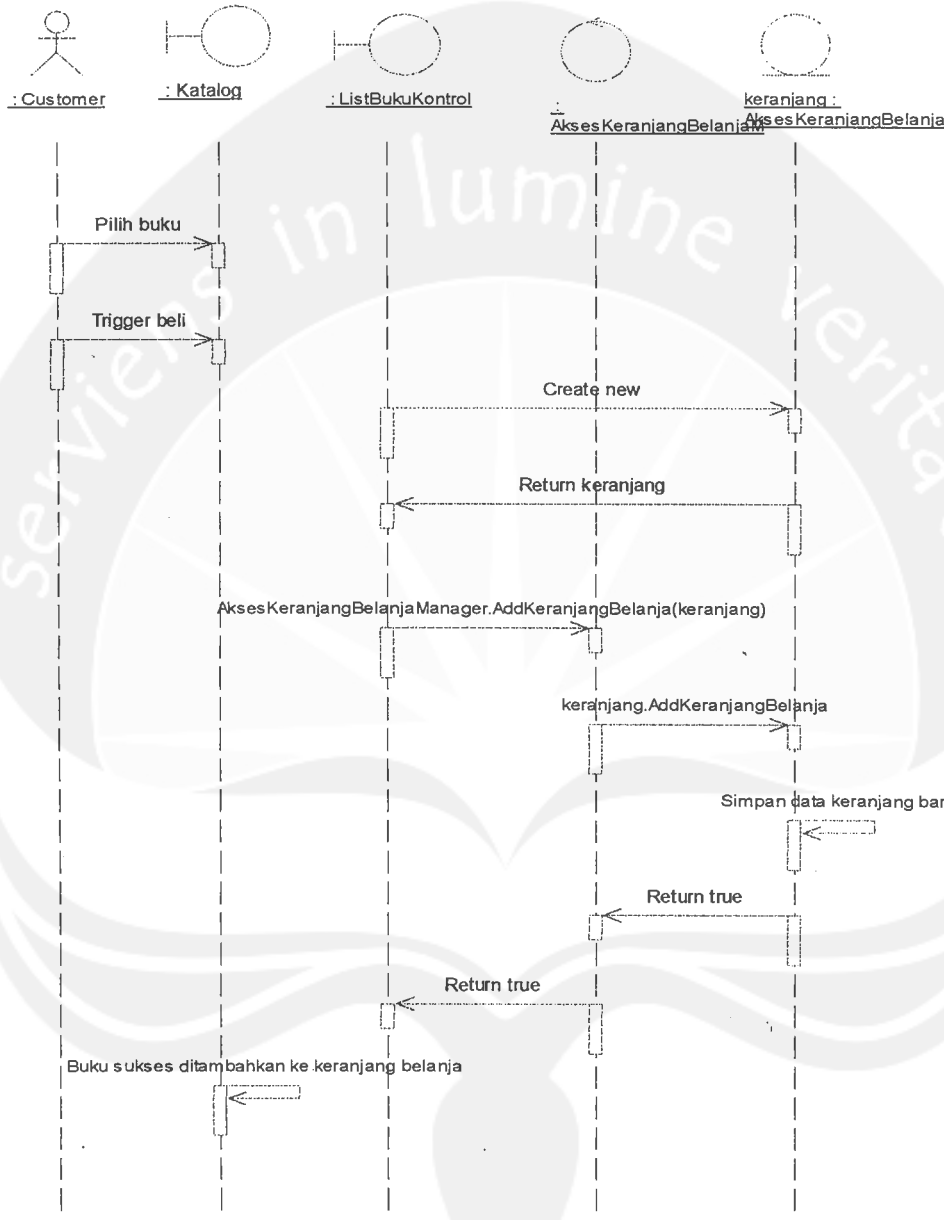
Gambar 2.45 Design Sequence Diagram : Use Case Pencarian Catalog

Flow of events :

1. Customer menginputkan kunci pencarian pada antarmuka `SearchUI` kemudian melakukan trigger cari buku.
2. `SearchUI` melalui `ListBukuUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
3. `AksesCatalog` mereturnkan objek *catalog* ke kontrol `ListBukuUI`.
4. `ListBukuUI` memanggil fungsi *AksesCatalogManager.Search (catalog)* dari `AksesCatalogManager`.
5. `AksesCatalogManager` memanggil fungsi *Search* dari `AksesCatalog`.
6. `AksesCatalog` mencari data di basis data.
7. `AksesCatalog` mereturnkan data table ke `AksesCatalogManager`.
8. `AksesCatalogManager` mereturnkan data table ke `ListBukuUI`.
9. `SearchUI` menampilkan ke customer.

2.3.4 Use Case : Pengisian Keranjang Belanja

2.3.4.1 Tambah Data Buku ke Keranjang Belanja



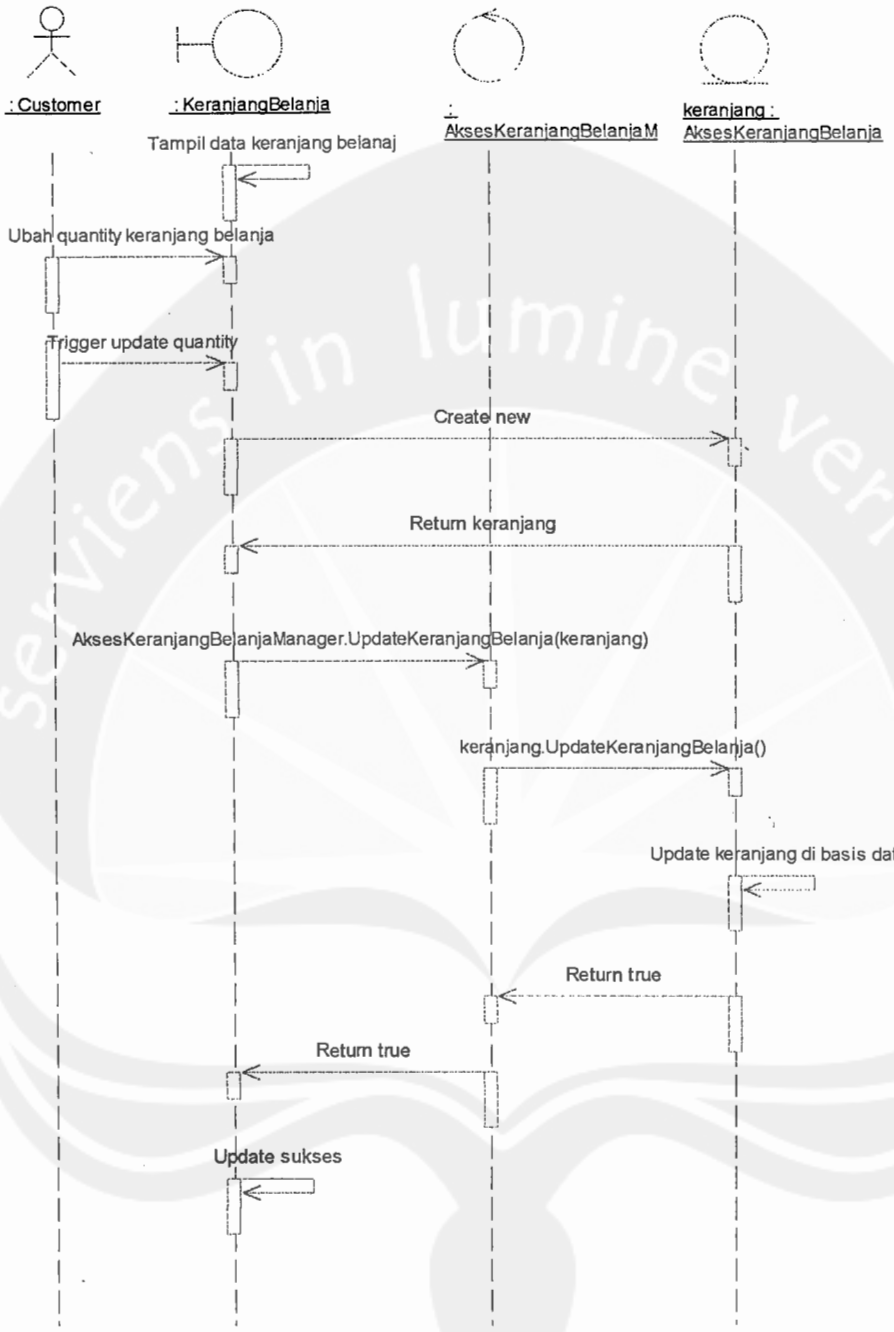
Gambar 2.46 Design Sequence Diagram : Use Case Pengisian Keranjang Belanja – Tambah Data Buku ke Keranjang Belanja

Flow of events :

1. `KatalogUI` menampilkan list buku ke customer.

2. Customer memilih buku kemudian melakukan trigger beli buku.
3. `KatalogUI` melalui `ListBukuUI` menciptakan new objek dari `AksesKeranjangBelanja` dengan nama *keranjang*.
4. `AksesKeranjangBelanja` mereturnkan objek *keranjang* ke kontrol `ListBukuUI`.
5. `ListBukuUI` memanggil fungsi *AksesKeranjangBelanjaManager.AddKeranjangBelanja(keranjang)* dari `AksesKeranjangBelanjaManager`.
6. `AksesKeranjangBelanjaManager` memanggil fungsi *AddKeranjangBelanja* dari `AksesKeranjangBelanja`.
7. `AksesKeranjangBelanja` menyimpan data ke basis data.
8. `AksesKeranjangBelanja` mereturnkan nilai *true* ke `AksesKeranjangBelanjaManager`.
9. `AksesKeranjangBelanjaManager` mereturnkan nilai *true* ke `ListBukuUI`.
10. `KatalogUI` menampilkan ke customer bahwa buku berhasil ditambahkan.

2.3.4.2 Ubah Jumlah Data Buku



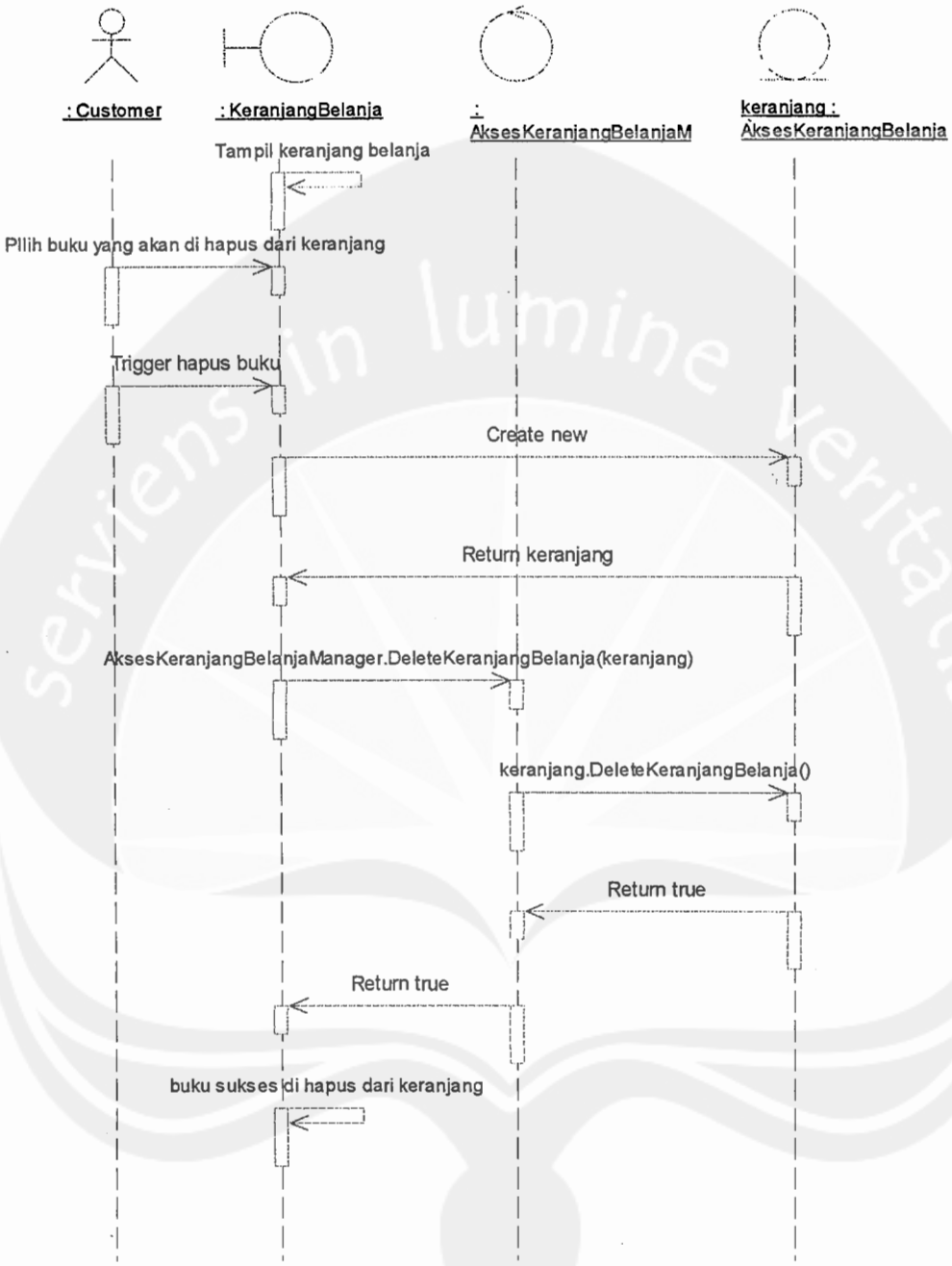
Gambar 2.47 Design Sequence Diagram : Use Case Pengisian Keranjang Belanja – Ubah Jumlah Data Buku

Flow of events :

1. `KeranjangBelanjaUI` menampilkan list data keranjang belanja ke customer.

2. Customer mengubah quantity item buku di keranjang belanja tersebut kemudian melakukan trigger update quantity.
3. `KeranjangBelanjaUI` menciptakan new objek dari `AksesKeranjangBelanja` dengan nama *keranjang*.
4. `AksesKeranjangBelanja` mereturnkan objek *keranjang* ke `KeranjangBelanjaUI`.
5. `KeranjangBelanjaUI` memanggil fungsi *AksesKeranjangBelanjaManager.UpdateKeranjangBelanja(keranjang)* dari `AksesKeranjangBelanjaManager`.
6. `AksesKeranjangBelanjaManager` memanggil fungsi *UpdateKeranjangBelanja* dari `AksesKeranjangBelanja`.
7. `AksesKeranjangBelanja` menyimpan data keranjang yang telah diupdate ke basis data.
8. `AksesKeranjangBelanja` mereturnkan nilai *true* ke `AksesKeranjangBelanjaManager`.
9. `AksesKeranjangBelanjaManager` mereturnkan nilai *true* ke `KeranjangBelanjaUI`.
10. `KeranjangBelanjaUI` menampilkan ke customer bahwa update berhasil.

2.3.4.3 Hapus Data Buku dari keranjang belanja



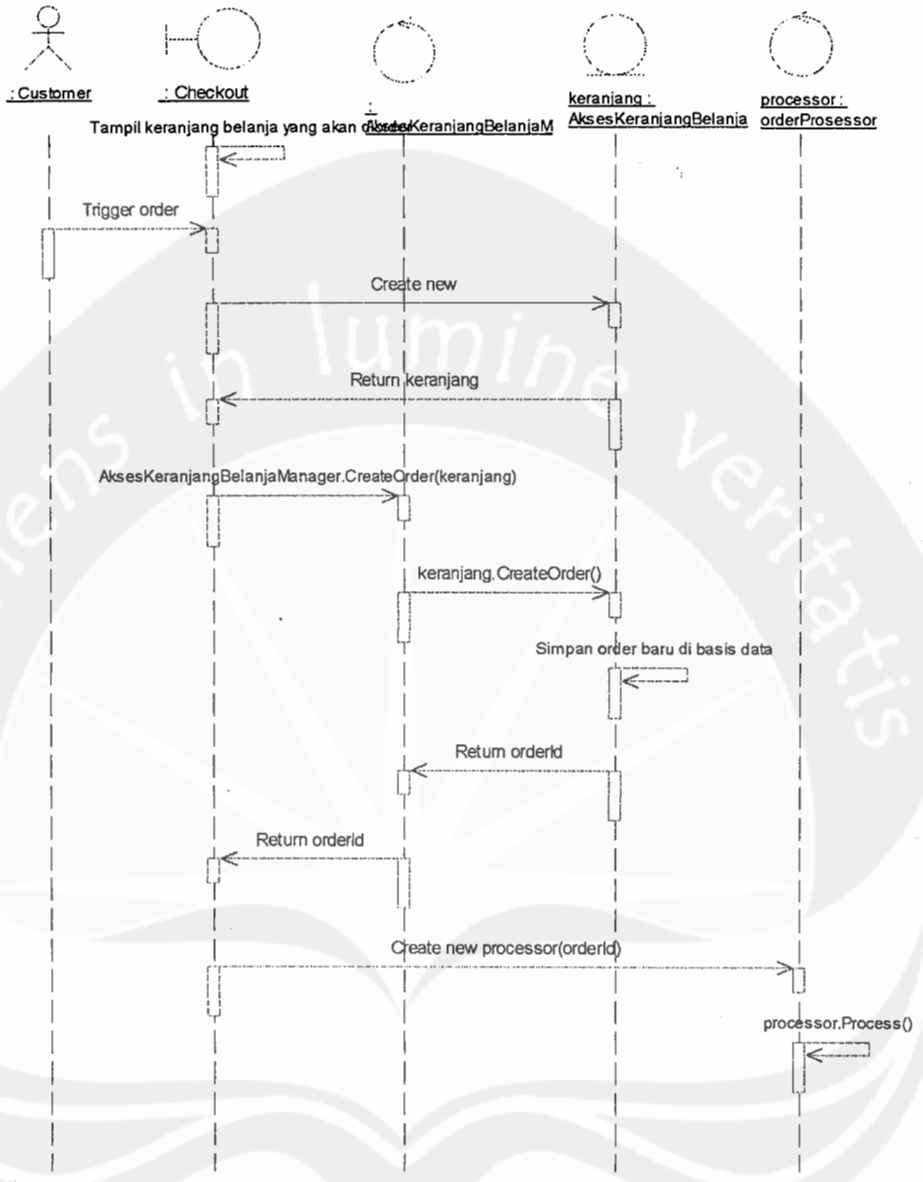
Gambar 2.48 Design Sequence Diagram : Use Case Pengisian Keranjang Belanja – Hapus Data Buku dalam Keranjang Belanja

Flow of events :

1. `KeranjangBelanjaUI` menampilkan list data keranjang belanja ke customer.

2. Customer memilih buku di keranjang belanja tersebut yang akan dihapus kemudian melakukan trigger hapus buku .
3. `KeranjangBelanjaUI` menciptakan new objek dari `AksesKeranjangBelanja` dengan nama *keranjang*.
4. `AksesKeranjangBelanja` mereturnkan objek *keranjang* ke `KeranjangBelanjaUI`.
5. `KeranjangBelanjaUI` memanggil fungsi *AksesKeranjangBelanjaManager.DeleteKeranjangBelanja(keranjang)* dari `AksesKeranjangBelanjaManager`.
6. `AksesKeranjangBelanjaManager` memanggil fungsi *DeleteKeranjangBelanja* dari `AksesKeranjangBelanja`.
7. `AksesKeranjangBelanja` menyimpan data keranjang yang telah diupdate ke basis data.
8. `AksesKeranjangBelanja` mereturnkan nilai *true* ke `AksesKeranjangBelanjaManager`.
9. `AksesKeranjangBelanjaManager` mereturnkan nilai *true* ke `KeranjangBelanjaUI`.
10. `KeranjangBelanjaUI` menampilkan ke customer bahwa buku sukses dihapus dari basis data.

2.3.5 Use Case : Order



Gambar 2.49 Design Sequence Diagram : Use Case Order

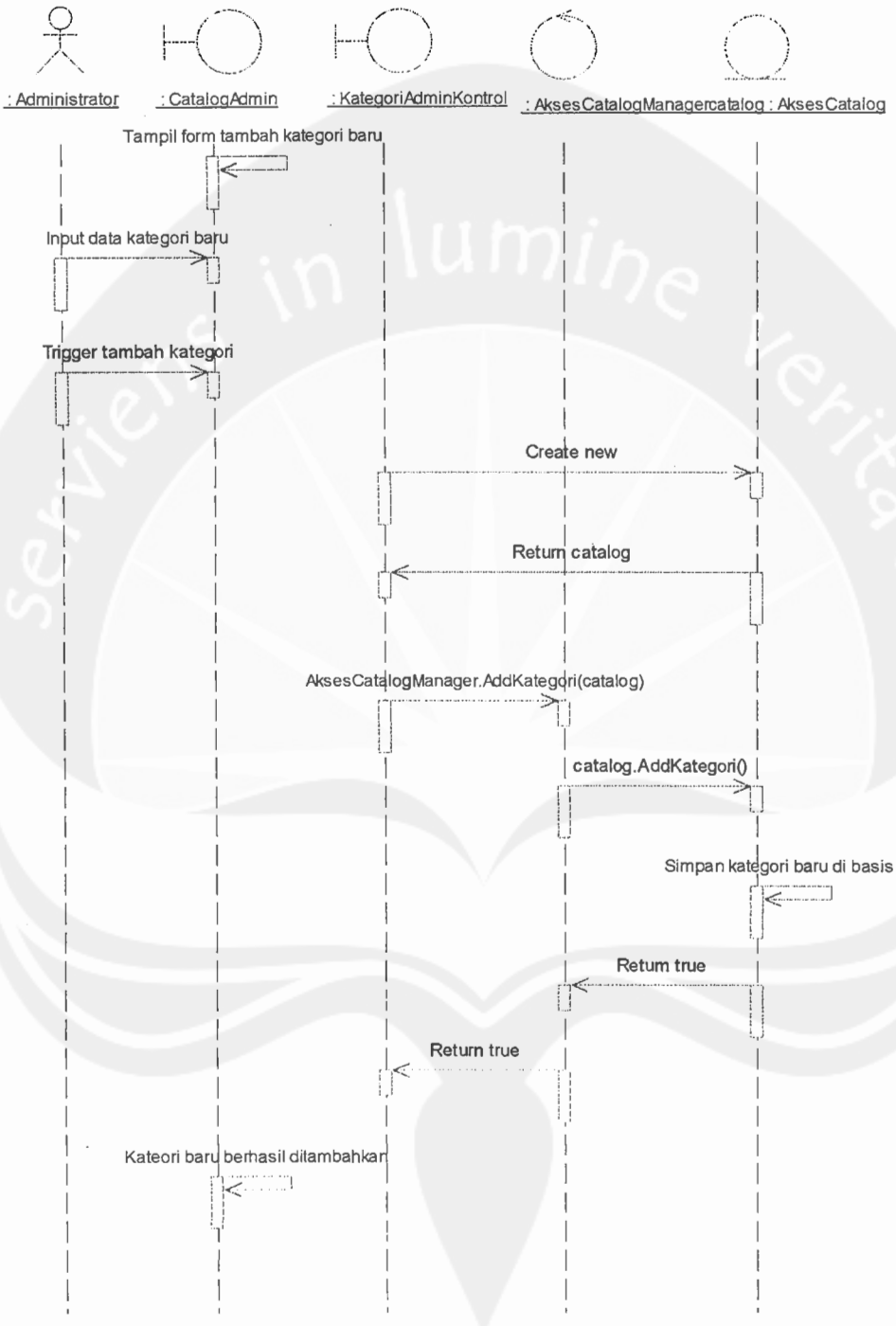
Flow of events :

1. `CheckoutUI` menampilkan data keranjang belanja yang siap diorder ke customer.
2. Customer mentrigger order.
3. `CheckoutUI` menciptakan new objek dari `AksesKeranjangBelanja` dengan nama *keranjang*.
4. `AksesKeranjangBelanja` mereturnkan objek *keranjang* ke `CheckoutUI`.

5. `CheckoutUI` memanggil fungsi *AksesKeranjangBelanjaManager.CreateOrder(keranjang)* dari `AksesKeranjangBelanjaManager.`
6. `AksesKeranjangBelanjaManager` memanggil fungsi *Create Order* dari `AksesKeranjangBelanja.`
7. `AksesKeranjangBelanja` menyimpan data order baru ke basis data.
8. `AksesKeranjangBelanja` mereturnkan *order Id* ke `AksesKeranjangBelanjaManager.`
9. `AksesKeranjangBelanjaManager` mereturnkan *order Id* ke `CheckoutUI.`
10. `CheckoutUI` menciptakan new objek dari `orderProcessor` dengan nama processor menggunakan parameter order Id.
11. `orderProcessor` memanggil fungsi *process* untuk melakukan proses order.

2.3.6 Use Case : Pengelolaan Catalog

2.3.6.1 Use Case : Tambah Kategori Baru

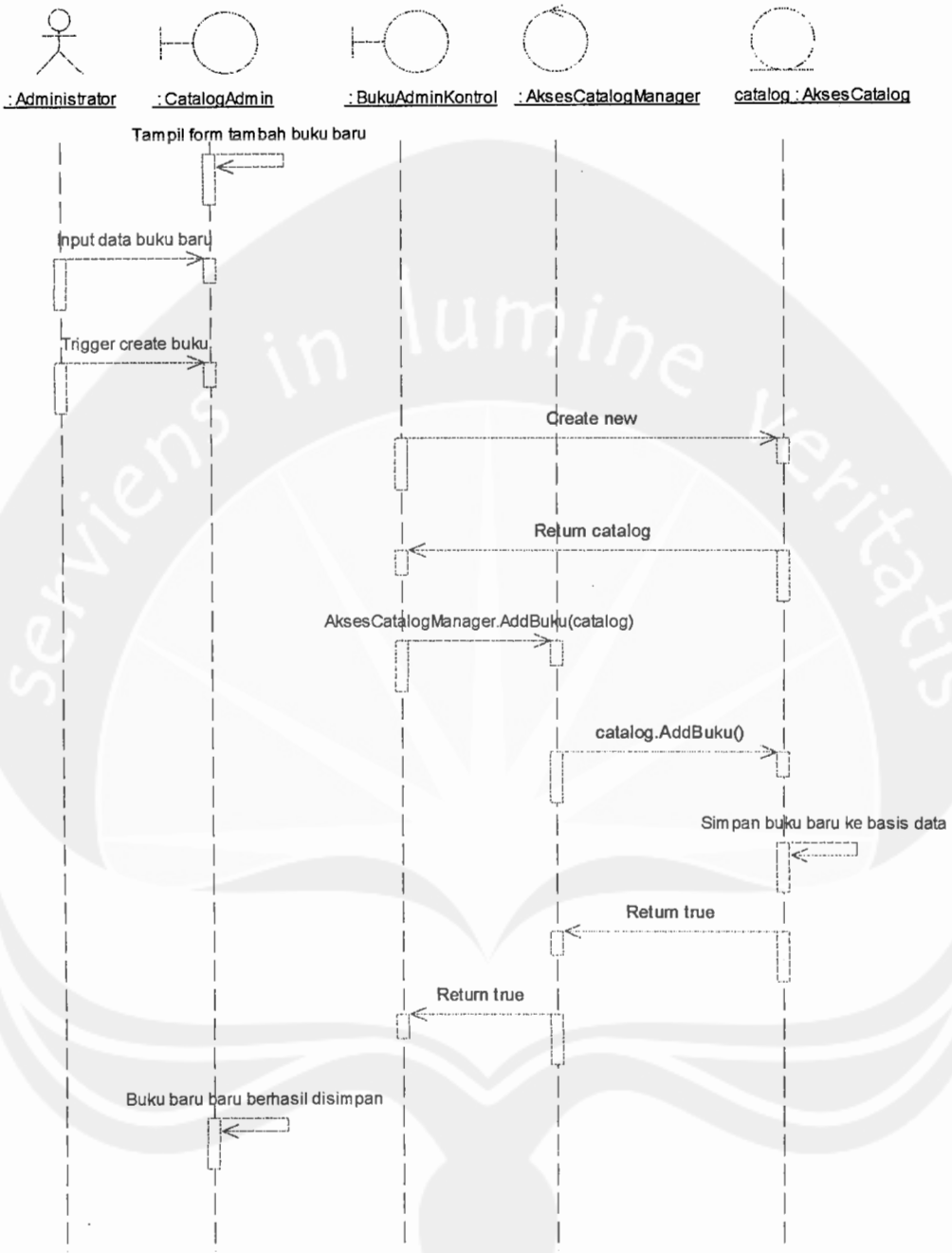


Gambar 2.50 Design Sequence Diagram : Use Case Pengelolaan Catalog – Tambah Kategori Baru

Flow of events :

1. `CatalogAdminUI` menampilkan form tambah kategori baru ke administrator.
2. Administrator menginputkan data kategori baru kemudian melakukan trigger tambah kategori.
3. `CatalogAdminUI` melalui kontrol `KategoriAdminUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
4. `AksesCatalog` mereturnkan objek *catalog* ke `KategoriAdminUI`.
5. `KategoriAdminUI` memanggil fungsi *AksesCatalogManager.AddKategori(catalog)* dari `AksesCatalogManager`.
6. `AksesCatalogManager` memanggil fungsi *AddKategori* dari `AksesCatalog`.
7. `AksesCatalog` menyimpan data kategori baru ke basis data.
8. `AksesCatalog` mereturnkan nilai *true* ke `AksesCatalogManager`.
9. `AksesCatalogManager` mereturnkan nilai *true* ke `KatalogUI`.
10. `CatalogAdminUI` menampilkan pesan bahwa kategori berhasil ditambahkan.

2.3.6.2 Use Case : Tambah Data Buku Baru



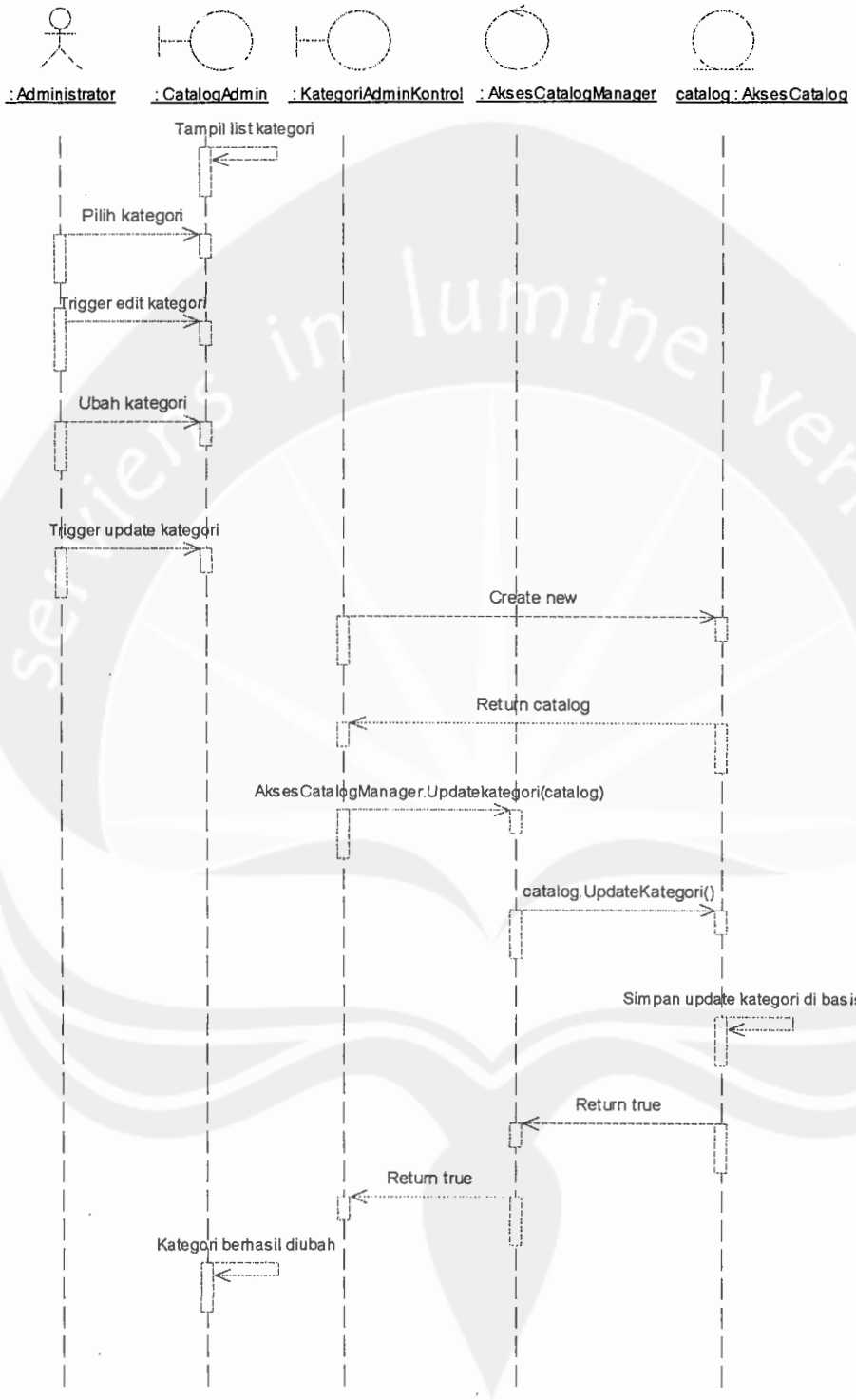
Gambar 2.51 Design Sequence Diagram : Use Case Pengelolaan Catalog – Tambah Data Buku Baru

Flow of events :

1. `CatalogAdminUI` menampilkan form tambah buku baru ke administrator.

2. Administrator menginputkan data buku baru kemudian melakukan trigger create buku.
3. `CatalogAdminUI` melalui kontrol `BukuAdminUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
4. `AksesCatalog` mereturnkan objek *catalog* ke `BukuAdminUI`.
5. `BukuAdminUI` memanggil fungsi *AksesCatalogManager.AddBuku(catalog)* dari `AksesCatalogManager`.
6. `AksesCatalogManager` memanggil fungsi *AddBuku* dari `AksesCatalog`.
7. `AksesCatalog` menyimpan data buku baru ke basis data.
8. `AksesCatalog` mereturnkan nilai *true* ke `AksesCatalogManager`.
9. `AksesCatalogManager` mereturnkan nilai *true* ke `BukuAdminUI`.
10. `CatalogAdminUI` menampilkan pesan bahwa buku baru berhasil disimpan.

2.3.6.3 Use Case : Ubah kategori

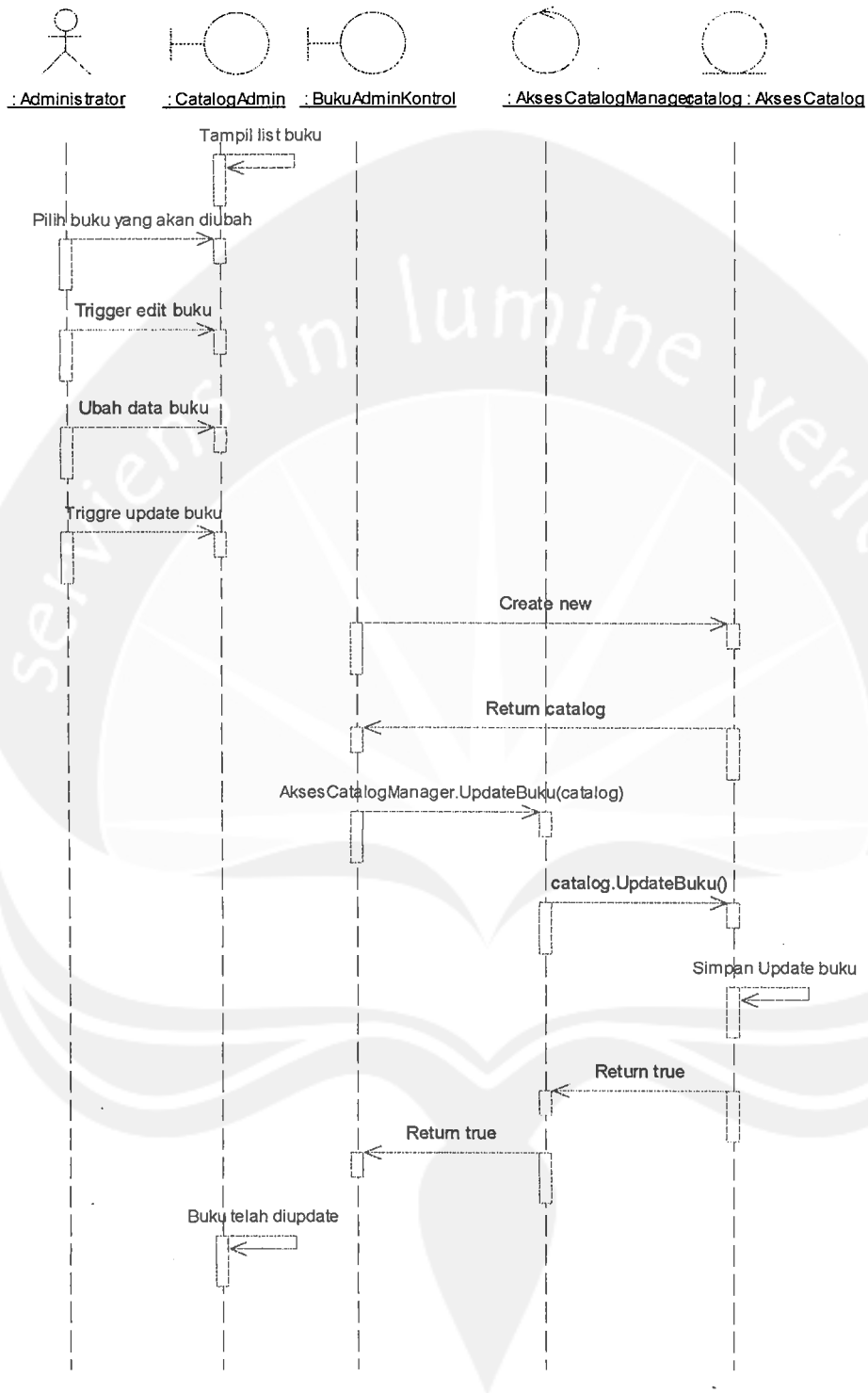


Gambar 2.52 Design Sequence Diagram : Use Case Pengelolaan Catalog – Ubah Kategori

Flow of events :

1. `CatalogAdminUI` menampilkan list kategori ke administrator.
2. Administrator memilih kategori kemudian melakukan trigger edit kategori.
3. Administrator mengubah data kategori kemudian melakukan trigger update kategori.
4. `CatalogAdminUI` melalui kontrol `KategoriAdminUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
5. `AksesCatalog` mereturnkan objek *catalog* ke `KatalogAdminUI`.
6. `KatalogAdminUI` memanggil fungsi *AksesCatalogManager.UpdateKategori(catalog)* dari `AksesCatalogManager`.
7. `AksesCatalogManager` memanggil fungsi *UpdateKategori* dari `AksesCatalog`.
8. `AksesCatalog` menyimpan data kategori yang telah diupdate ke basis data.
9. `AksesCatalog` mereturnkan nilai *true* ke `AksesCatalogManager`.
10. `AksesCatalogManager` mereturnkan nilai *true* ke `KategoriAdminUI`.
11. `CatalogAdminUI` menampilkan pesan bahwa kategori berhasil diupdate.

2.3.6.4 Use Case : Ubah Data Buku

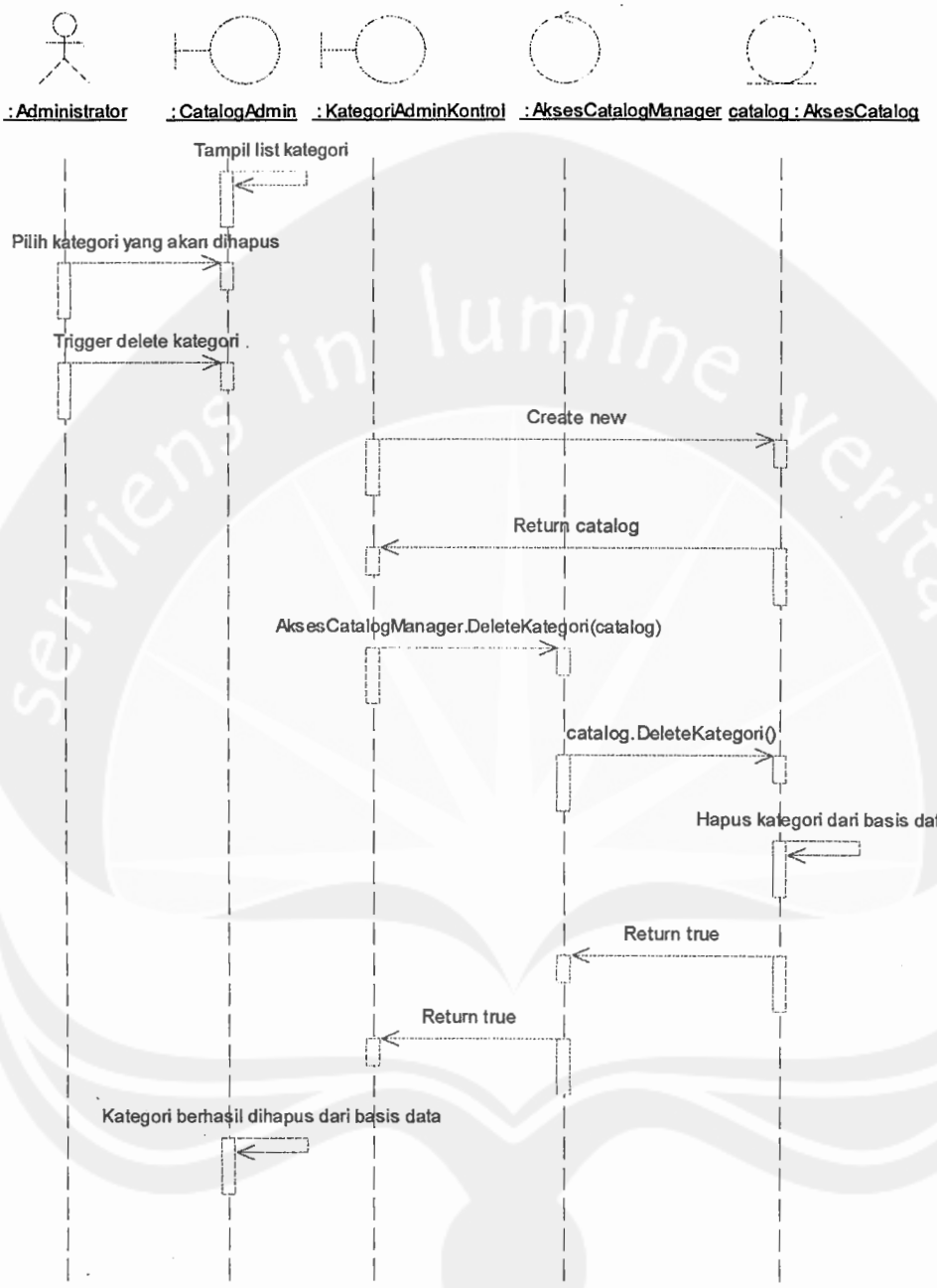


Gambar 2.53 Design Sequence Diagram : Use Case Pengelolaan Catalog – Ubah Data Buku

Flow of events :

1. `CatalogAdminUI` menampilkan list buku ke administrator.
2. Administrator memilih buku yang akan diubah kemudian melakukan trigger edit buku.
3. Administrator mengubah data buku kemudian melakukan trigger update buku.
4. `CatalogAdminUI` melalui kontrol `BukuAdminUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
5. `AksesCatalog` mereturnkan objek *catalog* ke `BukuAdminUI`.
6. `BukuAdminUI` memanggil fungsi *AksesCatalogManager.UpdateBuku(catalog)* dari `AksesCatalogManager`.
7. `AksesCatalogManager` memanggil fungsi *UpdateBuku* dari `AksesCatalog`.
8. `AksesCatalog` menyimpan data buku yang telah diupdate ke basis data.
9. `AksesCatalog` mereturnkan nilai *true* ke `AksesCatalogManager`.
10. `AksesCatalogManager` mereturnkan nilai *true* ke `BukuAdminUI`.
11. `CatalogAdminUI` menampilkan pesan bahwa buku berhasil diupdate.

2.3.6.5 Use Case : Hapus Kategori



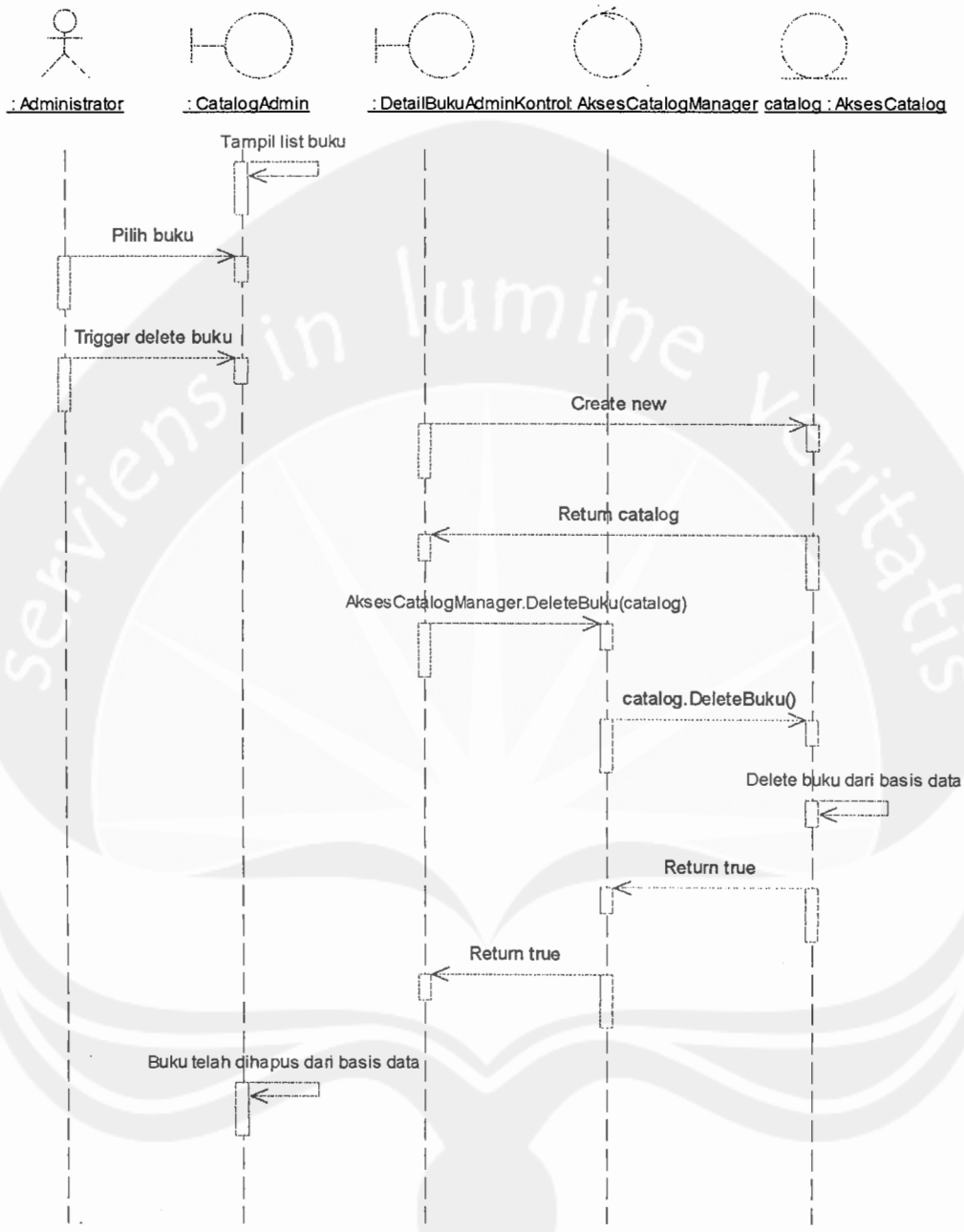
Gambar 2.54 Design Sequence Diagram : Use Case Pengelolaan Catalog – Hapus Kategori

Flow of events :

1. `CatalogAdminUI` menampilkan list kategori ke administrator.
2. Administrator memilih kategori yang akan dihapus kemudian melakukan trigger delete kategori.

3. `CatalogAdminUI` melalui kontrol `KategoriAdminUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
4. `AksesCatalog` mereturnkan objek *catalog* ke `KatalogAdminUI`.
5. `KatalogAdminUI` memanggil fungsi *AksesCatalogManager.DeleteKategori(catalog)* dari `AksesCatalogManager`.
6. `AksesCatalogManager` memanggil fungsi *DeleteKategori* dari `AksesCatalog`.
7. `AksesCatalog` menghapus data kategori dari basis data.
8. `AksesCatalog` mereturnkan nilai *true* ke `AksesCatalogManager`.
9. `AksesCatalogManager` mereturnkan nilai *true* ke `KategoriAdminUI`.
10. `CatalogAdminUI` menampilkan pesan bahwa kategori berhasil dihapus dari basis data.

2.3.6.6 Use Case : Hapus Data Buku



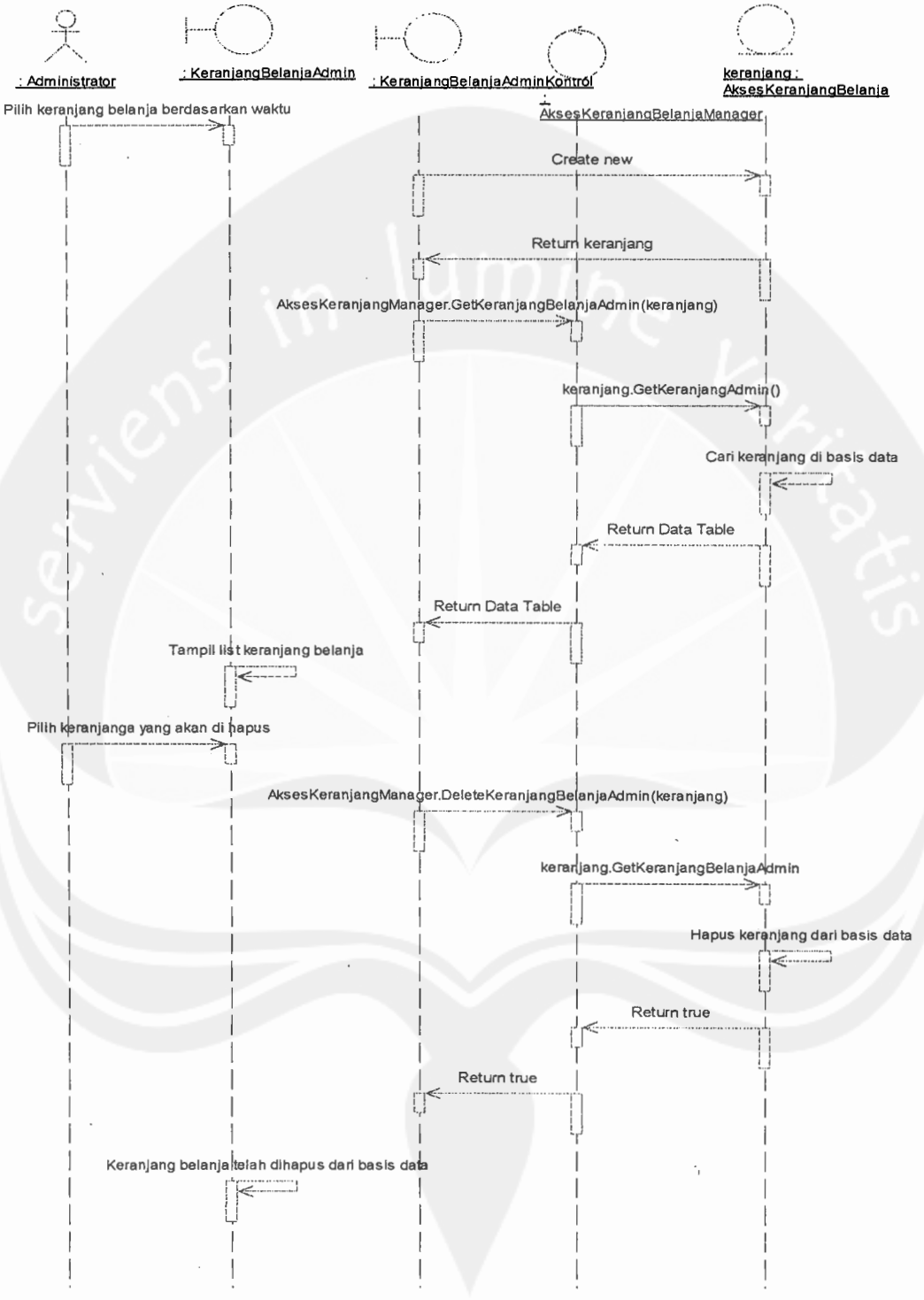
Gambar 2.55 Design Sequence Diagram : Use Case Pengelolaan Catalog – Hapus Data Buku

Flow of events :

1. `CatalogAdminUI` menampilkan list buku ke administrator.
2. Administrator memilih buku yang akan dihapus kemudian melakukan trigger delete kategori.

3. `CatalogAdminUI` melalui kontrol `DetailBukuAdminUI` menciptakan new objek dari `AksesCatalog` dengan nama *catalog*.
4. `AksesCatalog` mereturnkan objek *catalog* ke `DetailBukuAdminUI`.
5. `CatalogAdminUI` memanggil fungsi *AksesCatalogManager.DeleteBuku(catalog)* dari `AksesCatalogManager`.
6. `AksesCatalogManager` memanggil fungsi *DeleteBuku* dari `AksesCatalog`.
7. `AksesCatalog` menghapus data buku dari basis data.
8. `AksesCatalog` mereturnkan nilai *true* ke `AksesCatalogManager`.
9. `AksesCatalogManager` mereturnkan nilai *true* ke `DetailBukuAdminUI`.
10. `CatalogAdminUI` menampilkan pesan bahwa buku berhasil dihapus dari basis data.

2.3.7 Use Case : Pengelolaan Keranjang Belanja

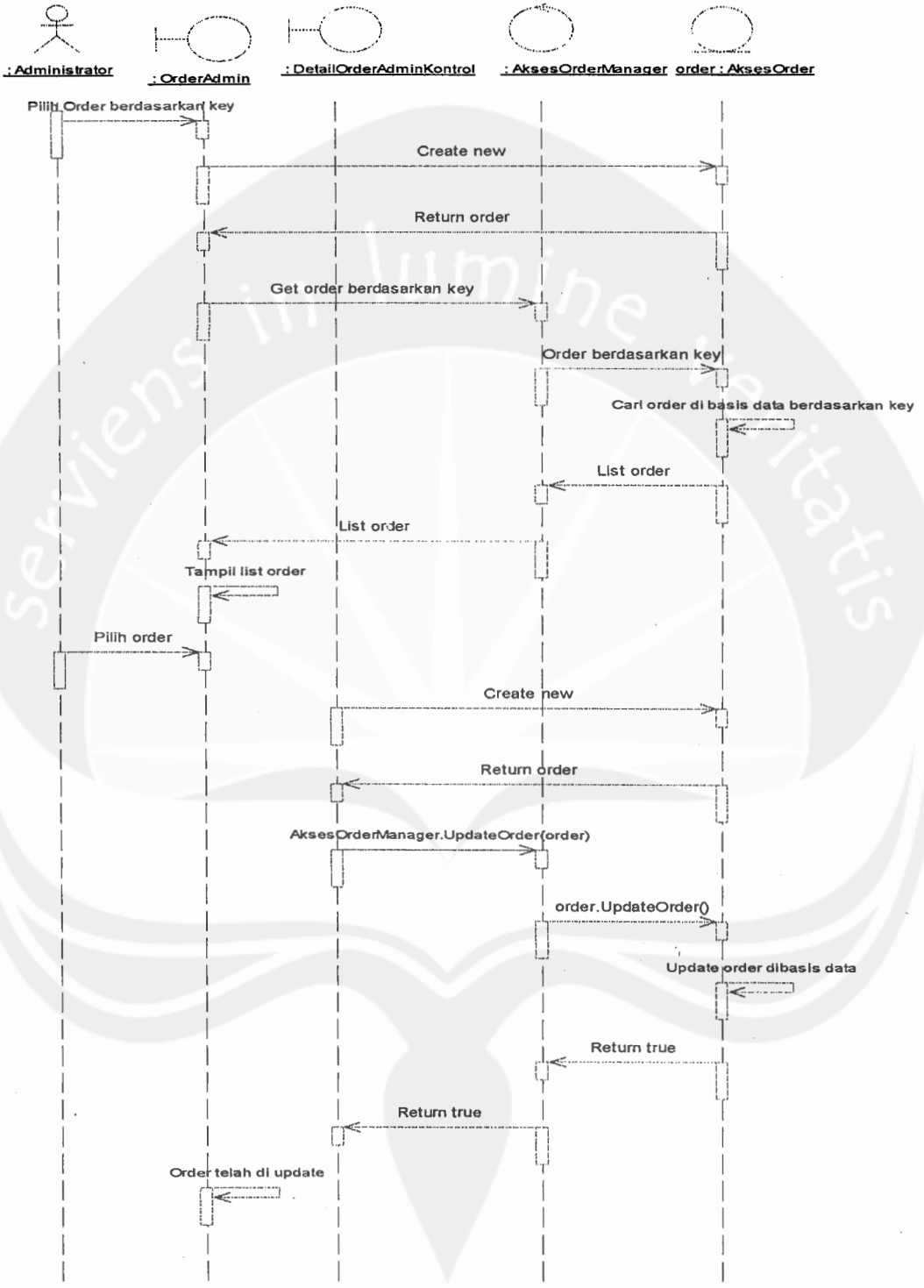


Gambar 2.56 Design Sequence Diagram : Use Case Pengelolaan Keranjang Belanja

Flow of events :

1. Administrator memilih limit waktu reorganisasi.
2. `KeranjangBelanjaAdminUI` melalui kontrol `KeranjangBelanjaAdminUI` menciptakan new objek dari `AksesKeranjangBelanja` dengan nama *keranjang*.
3. `AksesKeranjang` mereturnkan objek keranjang ke `KeranjangBelanjaAdminUI`.
4. `KeranjangBelanjaUI` memanggil fungsi *AksesKeranjangBelanjaManager.GetKeranjangBelanjaAdmin(keranjang)* dari `AksesKeranjangBelanjaManager`.
5. `AksesKeranjangBelanjaManager` memanggil fungsi GetKeranjangBelanjaAdmin dari `AksesKeranjangBelanja`.
6. `AksesKeranjangBelanja` mencari data keranjang di basis data.
7. `AksesKeranjangBelanja` mereturnkan Data Table ke `AksesKeranjangBelanjaManager`.
8. `AksesKeranjangBelanjaManager` mereturnkan Data table ke kontrol `KeranjangBelanjaAdminUI`.
9. `KeranjangBelanjaAdminUI` menampilkan list keranjang belanja ke administarator.
10. Administrator memilih keranjang yang akan dihapus.
11. `KeranjangBelanjaUI` memanggil fungsi *AksesKeranjangBelanjaManager.DeleteKeranjangBelanjaAdmin(keranjang)* dari `AksesKeranjangBelanjaManager`.
12. `AksesKeranjangBelanjaManager` memanggil fungsi DeleteKeranjangBelanjaAdmin dari `AksesKeranjangBelanja`.
13. `AksesKeranjangBelanja` menghapus keranjang dari basis data.
14. `AksesKeranjangBelanja` mereturnkan nilai true ke `AksesKeranjangBelanjaManager`.
15. `AksesKeranjangBelanjaManager` mereturnkan nilai true ke kontrol `KeranjangBelanjaAdminUI`.
16. `KeranjangBelanjaAdminUI` menampilkan pesan bahwa keranjang belanja berhasil di hapus dari basis data.

2.3.8 Use Case : Pengeloaan Order

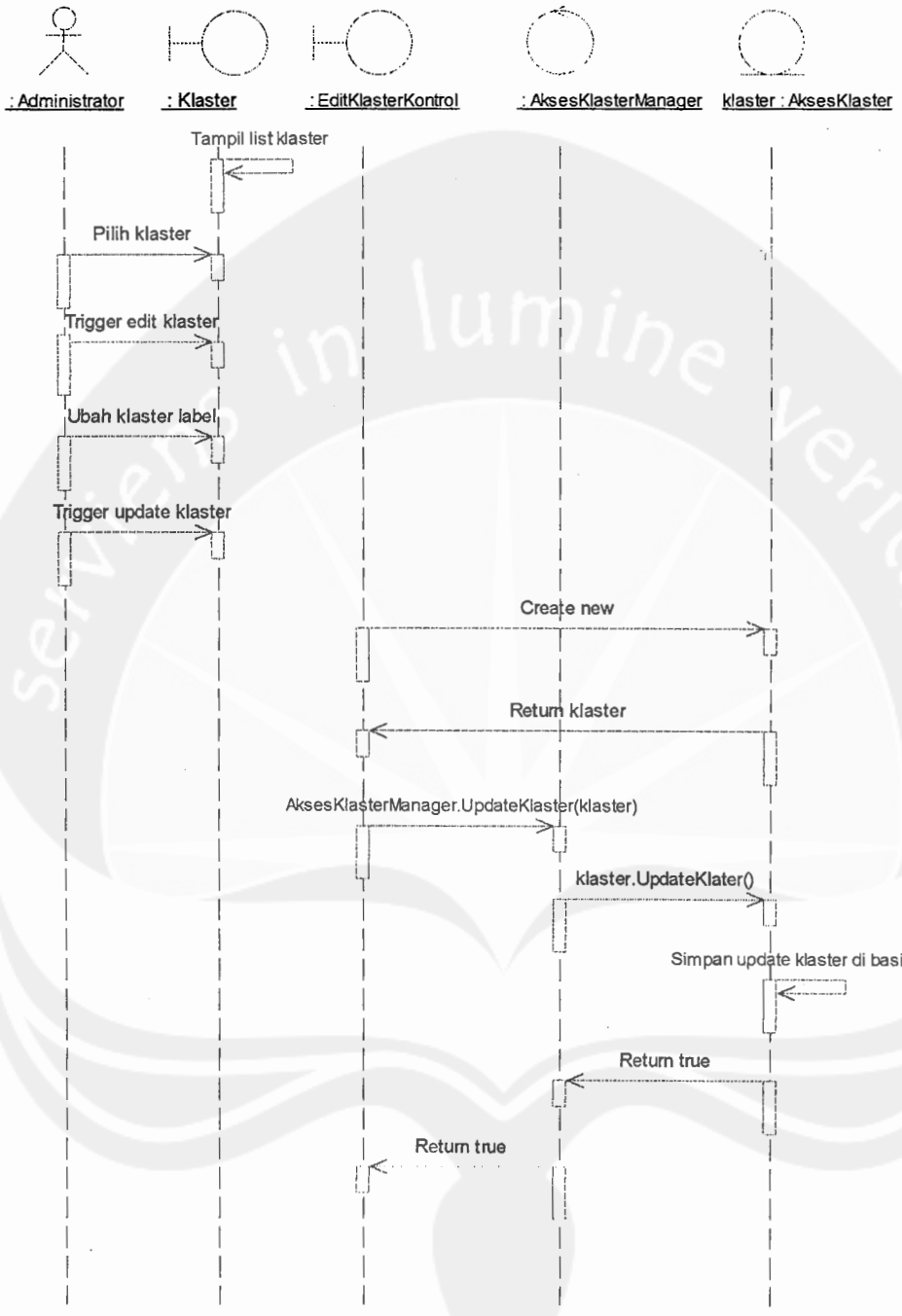


Gambar 2.57 Design Sequence Diagram : Use Case Pengelolaan Order

Flow of events :

1. Administrator memilih order berdasarkan key.
2. `OrderAdminUI` menciptakan new objek dari `AksesOrderManager` dengan nama *order*.
3. `AksesOrder` mereturnkan objek order ke `OrderAdminUI`.
4. `OrderAdminUI` memanggil fungsi *get order* berdasarkan key dari `AksesOrderManager`.
5. `AksesOrderManager` memanggil fungsi get order berdasarkan key dari `AksesOrder`.
6. `AksesOrder` mencari data order di basis data berdasarkan key.
7. `AksesOrder` mereturnkan list order ke `AksesOrderManager`.
8. `AksesOrderManager` mereturnkan list order ke kontrol `OrderAdminUI`.
9. `OrderAdminUI` menampilkan list keranjang belanja ke administarator.
10. Administrator memilih order.
11. `OrderAdminUI` melalui kontrol `DetailOrderAdminUI` menciptakan new objek dari `AksesOrderManager` dengan nama *order*.
12. `AksesOrder` mereturnkan objek order ke `DetailOrderAdminUI`.
13. `DetailOrderAdminUI` memanggil fungsi *AksesOrderManager.UpdateOrder(order)* dari `AksesOrderManager`.
14. `AksesOrderManager` memanggil fungsi *UpdateOrder* dari `AksesOrder`.
15. `AksesOrder` mengupdate data order di basis data.
16. `AksesOrder` mereturnkan nilai *true* ke `AksesOrderManager`.
17. `AksesOrderManager` mereturnkan nilai *true* ke kontrol `DetailOrderAdminUI`.
18. `OrderAdminUI` menampilkan pesan bahwa order telah diupdate.

2.3.9 Use Case : Pengeloaan Klaster



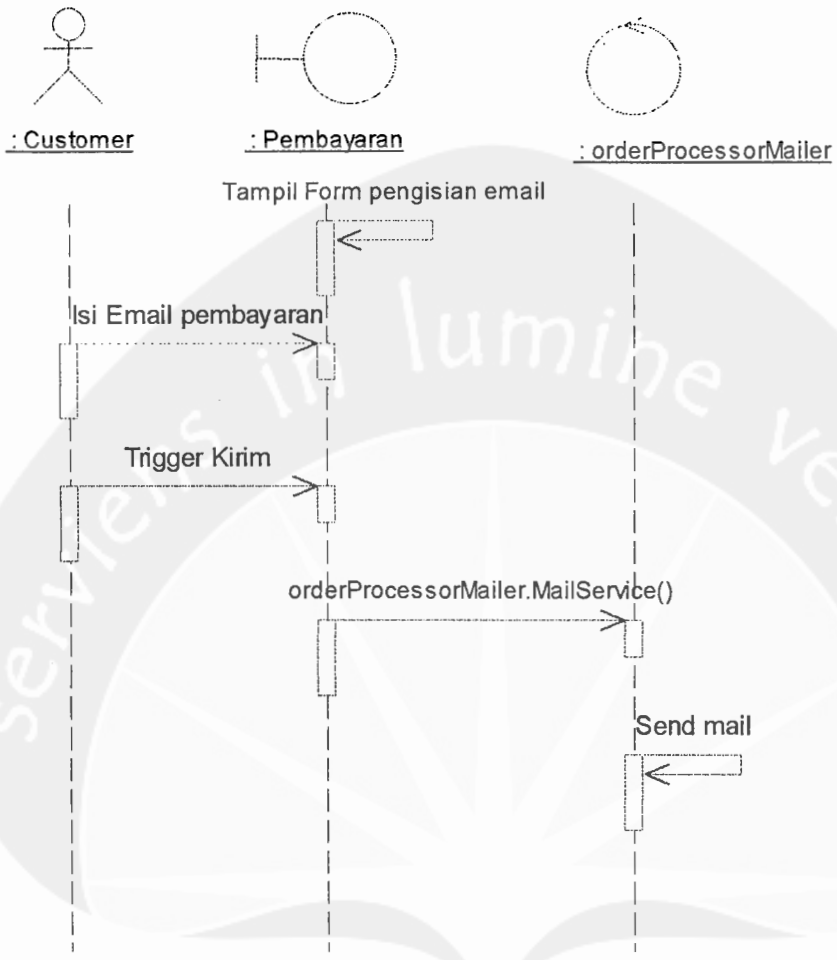
Gambar 2.58 Design Sequence Diagram : Use Case Pengelolaan Klaster

Flow of events :

1. `KlasterUI` menampilkan list klaster ke administrator.

2. Administrator memilih klaster kemudian melakukan trigger edit klaster.
3. Administrator mengubah label klaster kemudian melakukan trigger update klaster.
4. `KlasterUI` melalui kontrol `EditKlasterUI` menciptakan new objek dari `AksesKlaster` dengan nama *klaster*.
5. `AksesCatalog` mereturnkan objek *catalog* ke `BukuAdminUI`.
6. `EditKlasterUI` memanggil fungsi *AksesKlasterManager.UpdateKlaster(klaster)* dari `AksesKlasterManager`.
7. `AksesKlasterManager` memanggil fungsi *UpdateKlaster* dari `AksesKlaster`.
8. `AksesKlaster` menyimpan data klaster yang telah diupdate ke basis data.
9. `AksesKalster` mereturnkan nilai *true* ke `AksesKlasterManager`.
10. `AksesKlasterManager` mereturnkan nilai *true* ke `EditKlasterUI`.

2.3.10 Use Case : Notifikasi Pembayaran

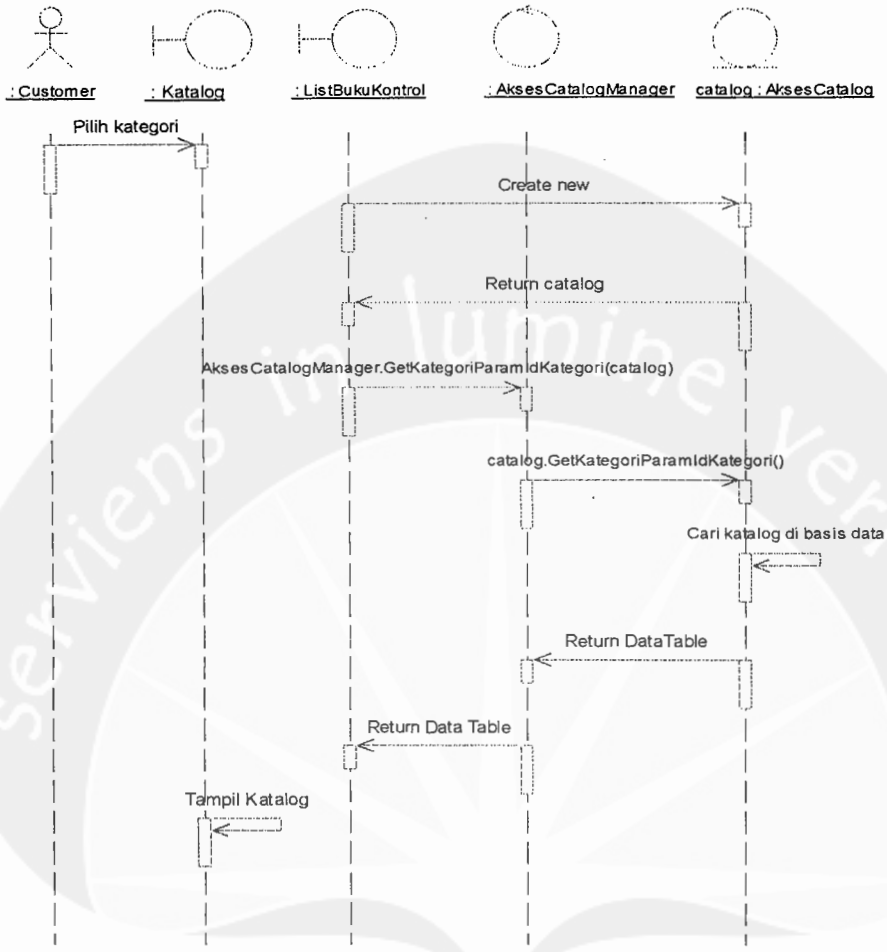


Gambar 2.59 Design Sequence Diagram : Use Case Notifikasi Pembayaran

Flow of events :

1. `PembayaranUI` menampilkan Form pengisian emaike administrator.
2. Administrator mengisi email kemudian melakukan trigger kirim email.
3. `PembayaranUI` memanggil fungsi *orderProcessorMailer.MailService()* dari `OrderProcessorMailer`.
4. `OrderProcessorMailer` memanggil fungsi *send mail*.

2.3.11 Use Case : Tampil Informasi



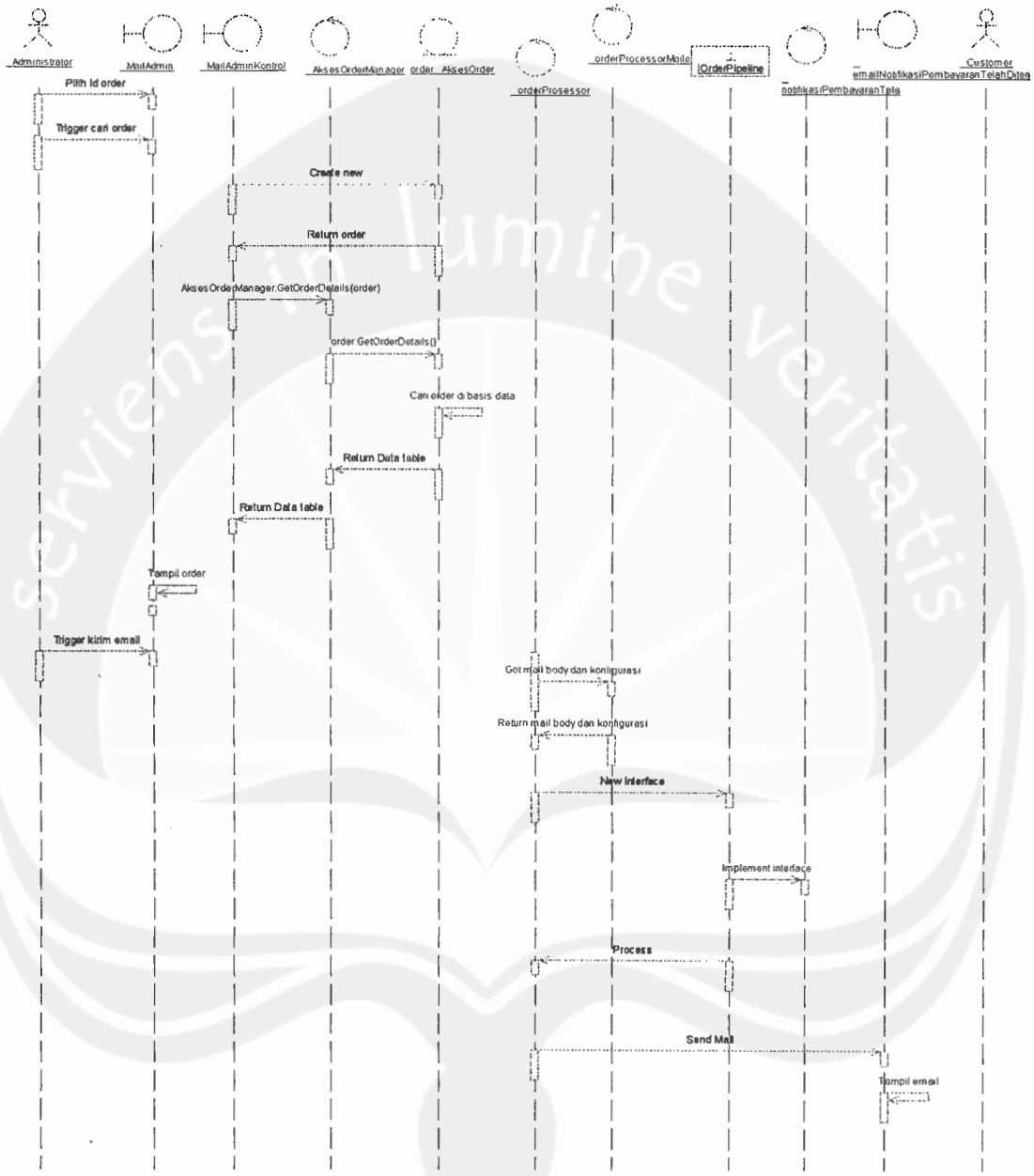
Gambar 2.60 Design Sequence Diagram : Use Case Klaster Tampil Informasi

Flow of events :

1. Customer memilih kategori.
2. `KatalogUI` melalui kontrol `ListBukuUI` menciptakan new objek dari `AksesCatalog` dengan nama *katalog*.
3. `AksesCatalog` mereturnkan objek *catalog* ke `ListBukuUI`.
4. `ListBukuUI` memanggil fungsi *AksesCatalogManager.GetKategoriParamIdKategori(katalog)* dari `AksesCatalogManager`.
5. `AksesCatalogManager` memanggil fungsi *GetKategoriParamIdKategori* dari `AksesCatalog`.
6. `AksesKlaster` mencari catalog di basis data.
7. `AksesCatalog` mereturnkan Data Table ke `AksesCatalogManager`.
8. `AksesCatalogManager` mereturnkan Data Table ke `ListBukuUI`.
9. `KatalogUI` menampilkan list catalog ke customer.

2.3.12 Use Case : Pengelolaan Email

2.3.12.1 Use Case : Email Notifikasi pembayaran telah diterima



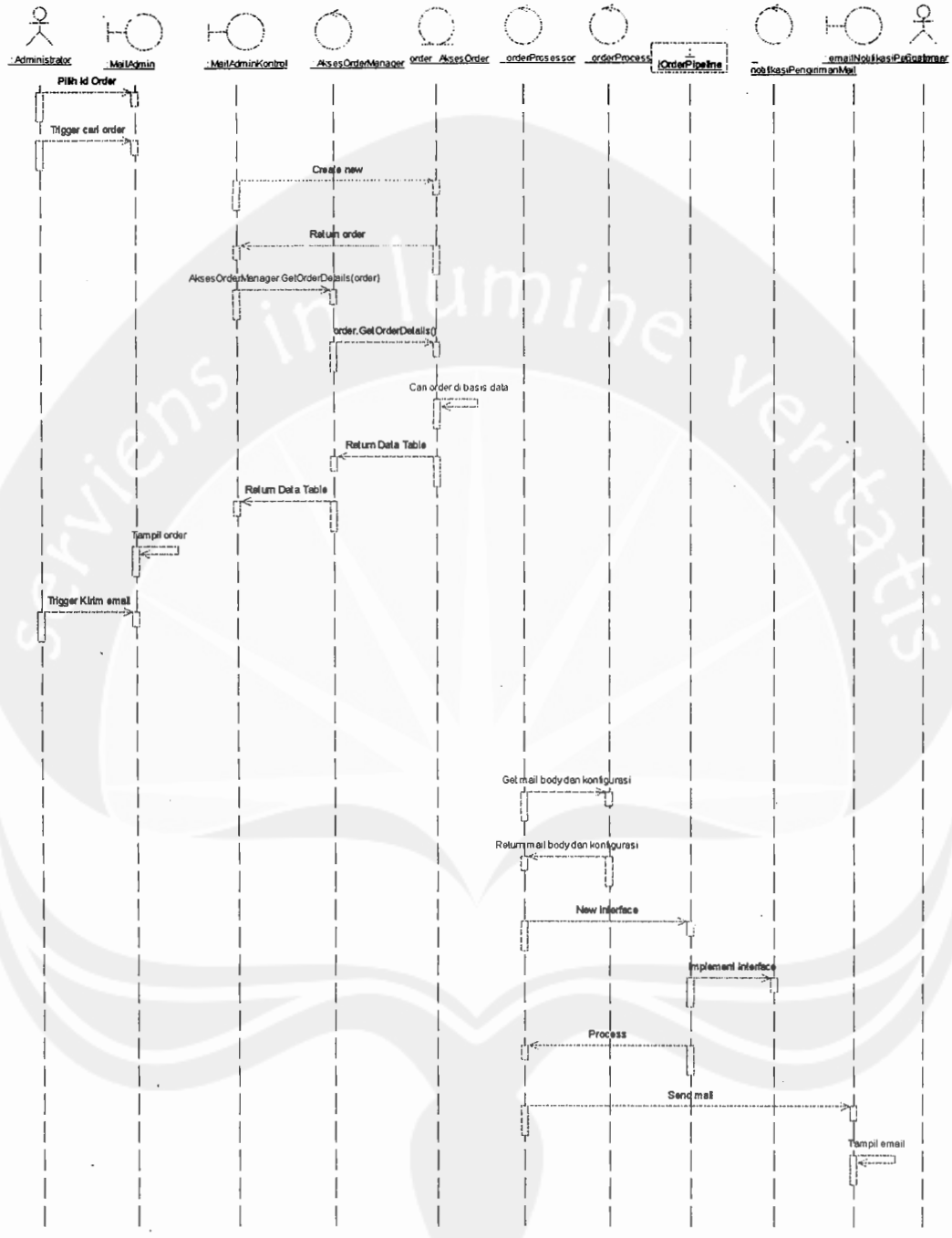
Gambar 2.61 Design Sequence Diagram : Use Case Pengelolaan Email – Email Notifikasi Pembayaran telah diterima

Flow of events :

1. Administrator memilih Id Order kemudian mentrigger cari order.

2. `MailAdminUI` melalui kontrol `MailAdminUI` menciptakan new objek dari `AksesOrder` dengan nama *order*.
3. `AksesOrder` mereturnkan objek *order* ke `MailAdminUI`.
4. `MailAdminUI` memanggil fungsi *AksesOrderManager.GetOrderDetails(order)* dari `AksesOrderManager`.
5. `AksesOrderManager` memanggil fungsi *GetOrderDetails* dari `AksesOrder`.
6. `AksesOrder` mencari order di basis data.
7. `AksesOrder` mereturnkan Data Table ke `AksesOrderManager`.
8. `AksesOrderManager` mereturnkan Data Table ke `MailAdminUI`.
9. `MailAdminUI` menampilkan list order ke administrator.
10. Administrator mentrigger kirim email.
11. `orderProcessor` memanggil fungsi get mail body dan konfigurasi dari `OrderProcessorMailer`.
12. `OrderProcessorMailer` mereturnkan mail body dan konfigurasi ke `orderProcessor`.
13. `orderProcessor` menciptakan new interface `iOrderPipeline`.
14. `iOrderPipeline` mengimplementasikan kelas `NotifikasiPembayaranTelahDiTerima`.
15. `iOrderPipeline` mereturnkan fungsi *Process* ke `orderProcessor`.
16. `orderProcessor` mengirim email ke customer.

2.3.12.2 Use Case : Email Notifikasi pengiriman



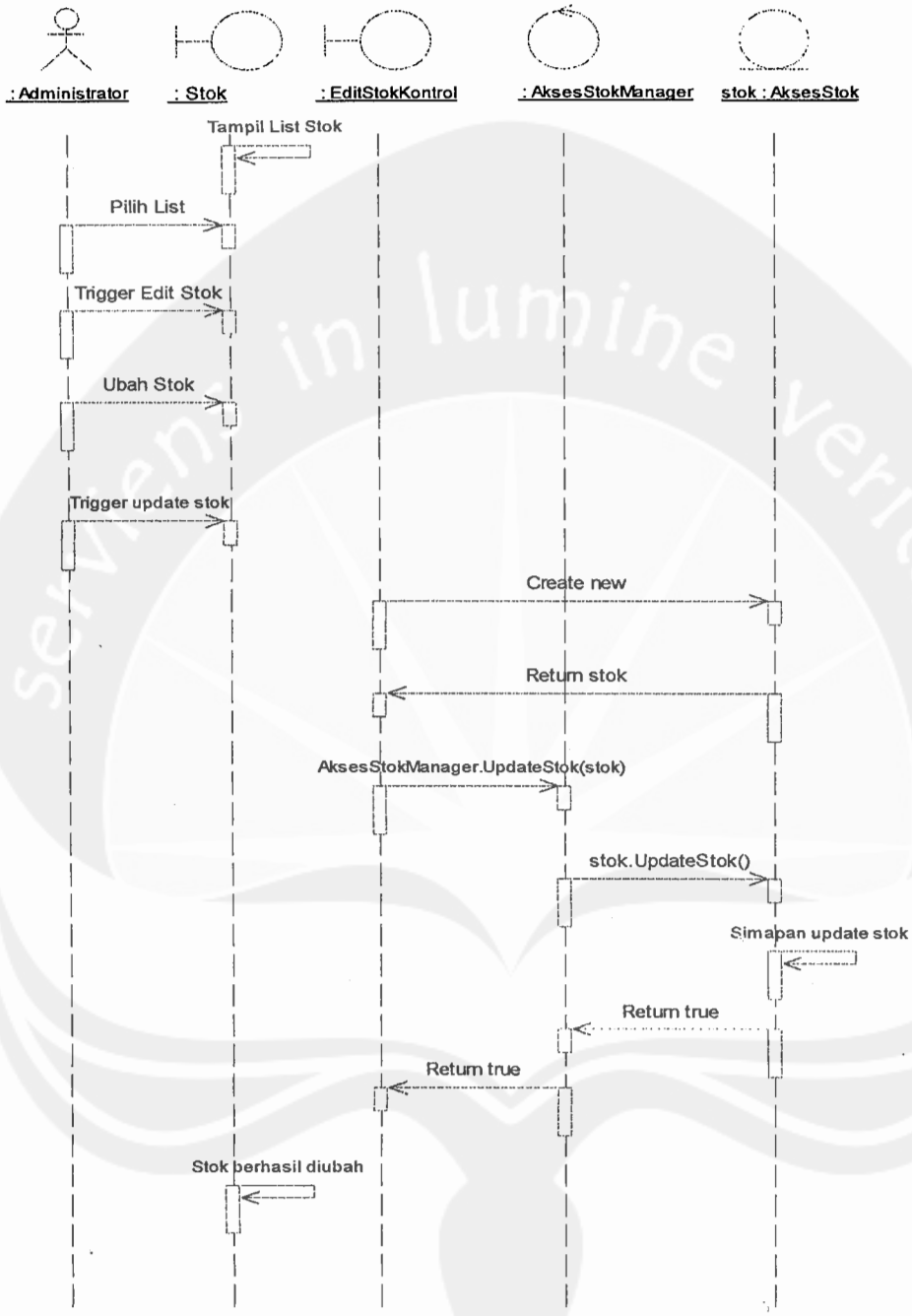
Gambar 2.62 Design Sequence Diagram : Use Case Pengelolaan Mail – Email Notifikasi Pengiriman

Flow of events :

1. Administrator memilih Id Order kemudian mentrigger cari order.

2. `MailAdminUI` melalui kontrol `MailAdminUI` menciptakan new objek dari `AksesOrder` dengan nama *order*.
3. `AksesOrder` mereturnkan objek *order* ke `MailAdminUI`.
4. `MailAdminUI` memanggil fungsi *AksesOrderManager.GetOrderDetails(order)* dari `AksesOrderManager`.
5. `AksesOrderManager` memanggil fungsi *GetOrderDetails* dari `AksesOrder`.
6. `AksesOrder` mencari order di basis data.
7. `AksesOrder` mereturnkan Data Table ke `AksesOrderManager`.
8. `AksesOrderManager` mereturnkan Data Table ke `MailAdminUI`.
9. `MailAdminUI` menampilkan list order ke administrator.
10. Administrator mentrigger kirim email.
11. `orderProcessor` memanggil fungsi get mail body dan konfigurasi dari `OrderProcessorMailer`.
12. `OrderProcessorMailer` mereturnkan mail body dan konfigurasi ke `orderProcessor`.
13. `orderProcessor` menciptakan new interface `iOrderPipeline`.
14. `iOrderPipeline` mengimplementasikan kelas `NotifikasiPengiriman`.
15. `iOrderPipeline` mereturnkan fungsi *Process* ke `orderProcessor`.
16. `orderProcessor` mengirim email ke customer.

2.3.13 Use Case : Pengelolaan Stok



Gambar 2.63 Design Sequence Diagram : Use Case Pengelolaan Stok

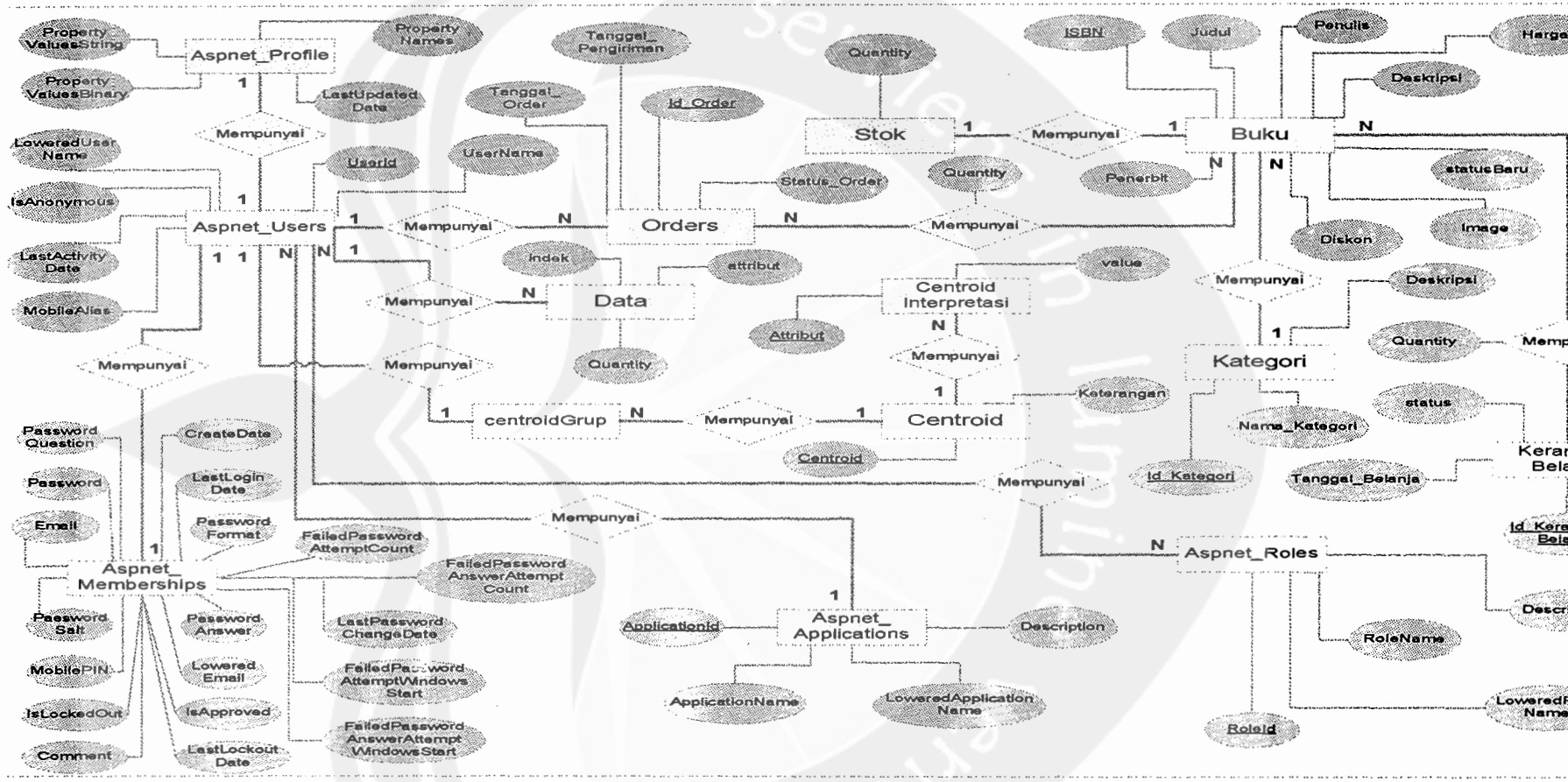
Flow of events :

1. `StokUI` menampilkan list stok ke administrator.
2. Administrator memilih stok yang akan diedit kemudian melakukan trigger edit.

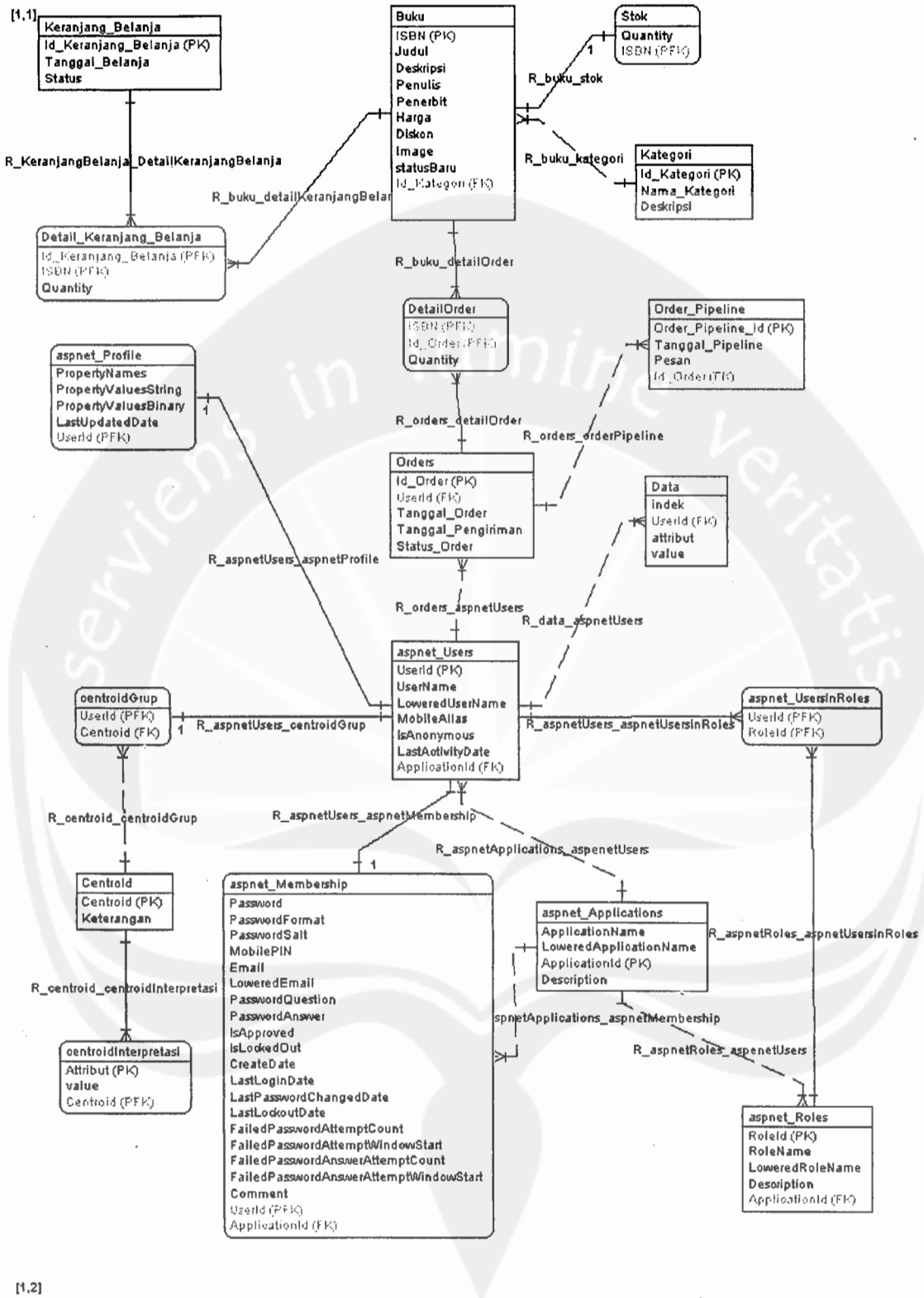
3. Administrator mengubah data stok kemudian melakukan trigger update stok.
4. `StokUI` melalui kontrol `EditStokUI` menciptakan new objek dari `AksesStok` dengan nama *stok*.
5. `AksesCatalogStok` mereturnkan objek stok ke `EditStokUI`.
6. `EditStokUI` memanggil fungsi *AksesStokManager.UpdateStok(stok)* dari `AksesStokManager`.
7. `AksesStokManager` memanggil fungsi *UpdateStok* dari `AksesStok`.
8. `AksesStok` menyimpan data klaster yang telah diupdate ke basis data.
9. `AksesStok` mereturnkan nilai *true* ke `AksesStokManager`.
10. `AksesStokManager` mereturnkan nilai *true* ke `EditStokUI`.
11. `StokUI` menampilkan pesan bahwa stok berhasil di ubah

3 Deskripsi Perancangan Persistent Data

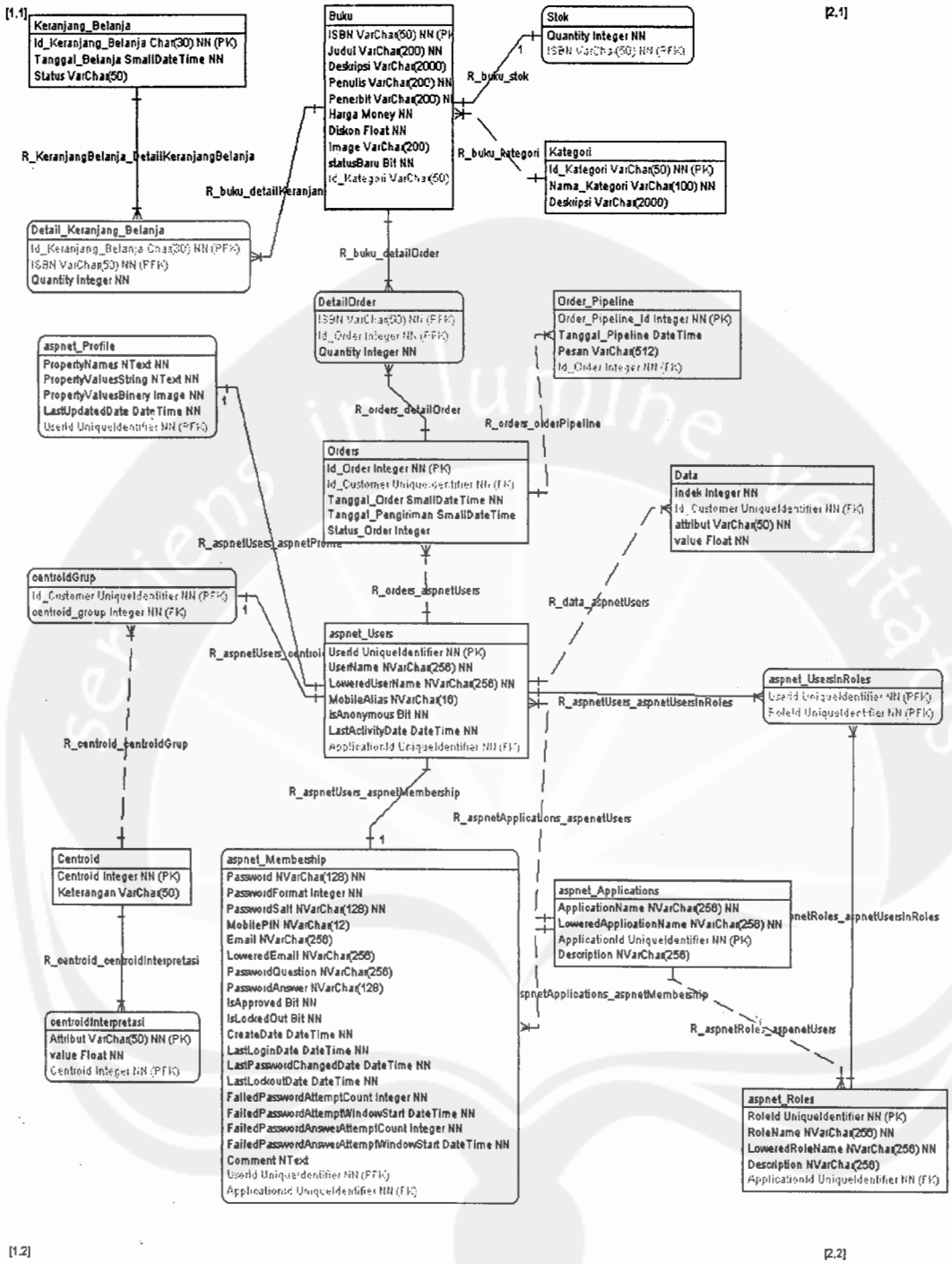
3.1 Basis Data



Gambar 3.1 Entity Relationship Siagram MERISKA



Gambar 3.2 Conceptual Data Model



Gambar 3.3 Physical Data Model

Tabel- tabel pada basis data disesuaikan dengan **Gambar 3.1**, **Gambar 3.2** dan **Gambar 3.3**. DBMS yang akan digunakan adalah SQL Server 2005.

3.1.1 Tabel Buku

Tabel 3.1 Tabel Buku

Nama	Tipe Data	NULL	Keterangan
ISBN	varchar(50)	No	Isbn buku, primary key
Judul	varchar(200)	No	Judul buku
Deskripsi	varchar(MAX)	Yes	Deskripsi tentang buku
Penulis	varchar(200)	No	Penulis buku
Penerbit	varchar(200)	No	Penerbit buku
Harga	money	No	Harga buku
Diskon	float	No	Diskon
Image	varchar(200)	Yes	Image buku
statusBaru	bit	No	Status dari buku. (baru atau tidak)
Id_Kategori	varchar(50)	No	Id dari kategori, foreign key

3.1.2 Tabel Kategori

Tabel 3.2 Tabel Kategori

Nama	Tipe Data	NULL	Keterangan
Id_Kategori	varchar(50)	No	Id kategori, primary key
Nama_Kategori	varchar(100)	No	Nama kategori
Deskripsi	varchar(MAX)	Yes	Deskripsi tentang kategori

3.1.3 Tabel Stok

Tabel 3.3 Tabel Stok

Nama	Tipe Data	NULL	Keterangan
ISBN	varchar(50)	No	Isbn buku, primary key dan foreign key
Quantity	int	No	Quantity stok dari buku

3.1.4 Tabel Keranjang Belanja

Tabel 3.4 Tabel Keranjang Belanja

Nama	Tipe Data	NULL	Keterangan
Id_Keranjang_Belanja	char(30)	No	Id keranjang belanja, primary key
Tanggal_Belanja	smalldatetime	No	Tanggal Belanja
Status	varchar(50)	Yes	Status dari keranjang belanja

3.1.5 Tabel Detail Keranjang Belanja

Tabel 3.5 Tabel Detail Keranjang Belanja

Nama	Tipe Data	NULL	Keterangan
Id_Keranjang_Belanja	char(30)	No	Id keranjang belanja, primary key dan foreign key
ISBN	varchar(50)	No	Isbn buku, primary key dan foreign key
Quantity	int	No	Quantity dari item di keranjang belanja

3.1.6 Tabel Orders

Tabel 3.6 Tabel Orders

Nama	Tipe Data	NULL	Keterangan
Id_Order	int	No	Id order, primary key
Tanggal_Order	smalldatetime	No	Tanggal transaksi order
Tanggal_Pengiriman	smalldatetime	Yes	Tanggal Pengiriman order
Id_Customer	uniqueidentifier	No	Id customer yang melakukan order, foreign key

Status_Order	int	Yes	Status order

3.1.7 Tabel Detail Order

Tabel 3.7 Tabel Detail Order

Nama	Tipe Data	NULL	Keterangan
Id_Order	int	No	Id order, primary key dan foreign key
ISBN	varchar(50)	No	Isbn buku, primary key dan foreign key
Quantity	int	No	Quantity dari item yang diorder

3.1.8 Tabel Order Pipeline

Tabel 3.8 Tabel Order Pipeline

Nama	Tipe Data	NULL	Keterangan
Order_Pipeline_Id	int	No	Id order pipeline, primary key
Id_Order	int	No	Id order, foreign key
Tanggal_Pipeline	datetime	No	Tanggal terjadinya setiap event proses order
Pesan	varchar(512)	No	Pesan dari setiap event proses order

3.1.9 Tabel Data

Tabel 3.9 Tabel Data

Nama	Tipe Data	NULL	Keterangan
indek	int	No	Indek dari data, primary key
Id_Customer	uniqueidentifier	No	Id customer, foreign key
attribut	varchar(50)	No	Attribut data

			(kategori)
value	float	No	Nilai dari attribute(rata-rata item yang dibeli berdasarkan kategori)

3.1.10 Tabel Centroid Interpretasi

Tabel 3.10 Tabel Centroid Interpretasi

Nama	Tipe Data	NULL	Keterangan
Centroid	int	No	Centroid klaster, primary key dan foreign key
Attribut	varchar(50)	No	Attribut klaster , primary key
value	float	No	Value klaster

3.1.11 Tabel Centroid

Tabel 3.11 Tabel Centroid

Nama	Tipe Data	NULL	Keterangan
Centroid	int	No	Centroid klaster, primary key
Keterangan	varchar(50)	Yes	Keterangan tentang dentroid klaster

3.1.12 Tabel Centroid Grup

Tabel 3.12 Tabel Centroid Grup

Nama	Tipe Data	NULL	Keterangan
centroid_group	int	No	Centroid klaster, foreign key
Id_Customer	uniqueidentifier	No	Id_customer,primary

			key dan foreign key

3.1.13 Tabel aspnet_Profile

Tabel 3.13 Tabel aspnet_Profile

Nama	Tipe Data	NULL	Keterangan
UserId	uniqueidentifier	No	Primarykey dan foreign key
PropertyNames	ntext	No	-
PropertyValuesString	ntext	No	-
PropertyValuesBinary	image	No	-
LastUpdatedDate	datetime	No	

3.1.14 Tabel aspnet_Membership

Tabel 3.14 Tabel aspnet_Membership

Nama	Tipe Data	NULL	Keterangan
ApplicationId	uniqueidentifier	No	Foreign key
UserId	uniqueidentifier	No	Primary key dan foreign key
Password	nvarchar(128)	No	-
PasswordFormat	int	No	-
PasswordSalt	nvarchar(128)	No	-
MobilePIN	nvarchar(16)	Yes	-
Email	nvarchar(256)	Yes	-
LoweredEmail	nvarchar(256)	Yes	-
PasswordQuestion	nvarchar(256)	Yes	-
PasswordAnswer	nvarchar(128)	Yes	-
IsApproved	bit	No	-
IsLockedOut	bit	No	-
CreateDate	datetime	No	-
LastLoginDate	datetime	No	-
LastPasswordChangedDate	datetime	No	-
LastLockoutDate	datetime	No	-

FailedPasswordAttemptCount	int	No	-
FailedPasswordAttemptWindowStart	datetime	No	-
FailedPasswordAnswerAttemptCount	int	No	-
FailedPasswordAnswerAttemptWindowStart	datetime	No	-
Comment	ntext	Yes	-

3.1.15 Tabel aspnet_Applications

Tabel 3.15 Tabel aspnet_Applications

Nama	Tipe Data	NULL	Keterangan
ApplicationId	uniqueidentifier	No	Primary key
LoweredApplicationName	nvarchar(256)	No	-
ApplicationName	nvarchar(256)	No	-
Description	nvarchar(256)	Yes	-

3.1.16 Tabel aspnet_Roles

Tabel 3.16 Tabel aspnet_Roles

Nama	Tipe Data	NULL	Keterangan
RoleId	uniqueidentifier	No	Primary key
ApplicationId	uniqueidentifier	No	Foreign key
RoleName	nvarchar(256)	No	-
LoweredRoleName	nvarchar(256)	No	-
Description	nvarchar(256)	Yes	-

3.1.17 Tabel aspnet_Users

Tabel 3.17 Tabel aspnet_Users

Nama	Tipe Data	NULL	Keterangan
UserId	uniqueidentifier	No	Primary key
ApplicationId	uniqueidentifier	No	Foreign

			key
UserName	nvarchar(256)	No	-
LoweredUserName	nvarchar(256)	No	-
MobileAlias	nvarchar(16)	Yes	-
IsAnonymous	bit	No	-
LastActivityDate	datetime	No	-

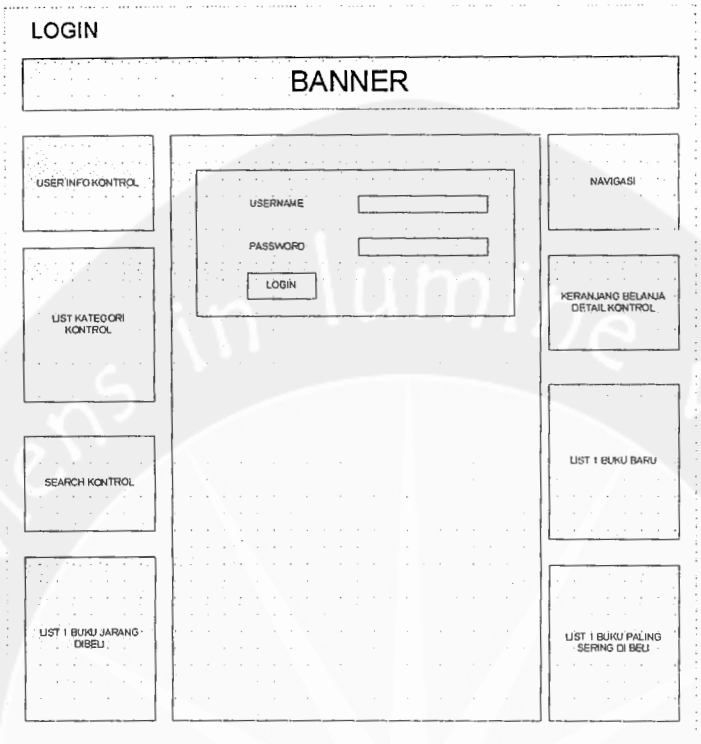
3.1.18 Tabel aspnet_UsersInRoles

Tabel 3.18 Tabel aspnet_UsersInRoles

Nama	Tipe Data	NULL	Keterangan
UserId	uniqueidentifier	No	Primary key dan foreign key
RoleId	uniqueidentifier	No	Primary key dan foreign key

4 Deskripsi Perancangan Antarmuka

4.1 Use Case : Login

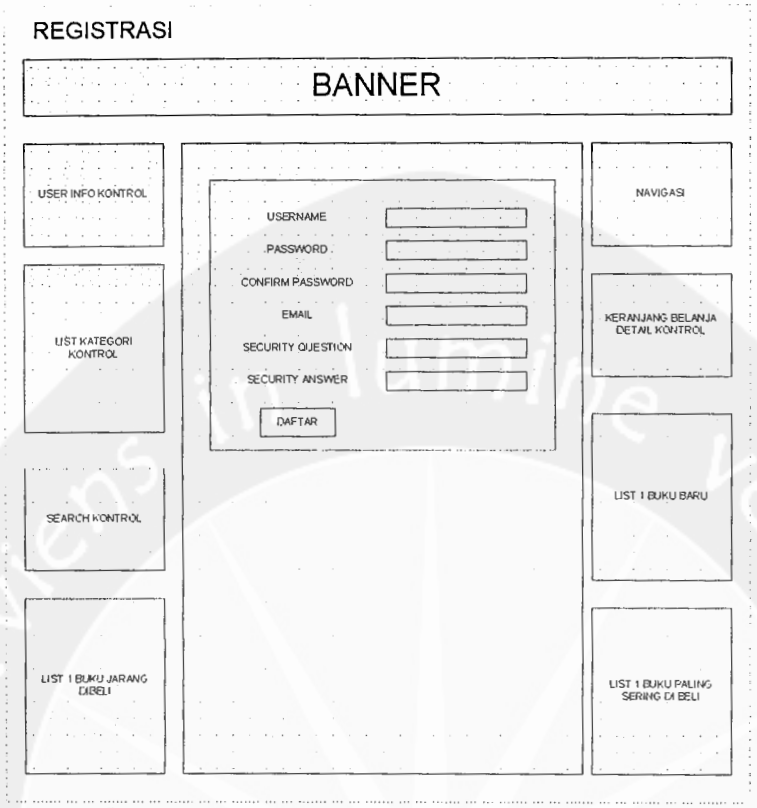


Gambar 4.1 Rancangan Antarmuka Form Login

Deskripsi

Antarmuka untuk proses login ke dalam sistem. Fasilitas login ini berlaku bagi user yang telah terdaftar sebagai customer maupun administrator. Form mengharuskan user menginputkan username dan password pada kontrol textbox. Pada saat tombol login ditekan, sistem akan melakukan pengecekan terhadap username dan password yang diinputkan dengan yang terdapat di basis data. Jika username dan password yang diinputkan cocok dengan yang terdapat di basis data maka user yang bersangkutan memperoleh akses masuk ke sistem sesuai dengan role-nya masing-masing.

4.2 Use Case : Registrasi

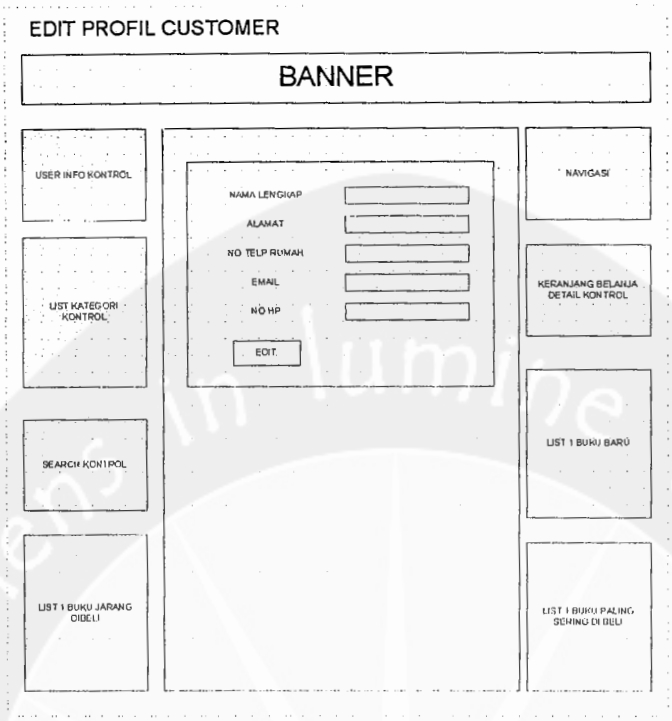


Gambar 4.2 Rancangan Antarmuka Form Registrasi

Deskripsi

Antarmuka untuk mendaftar menjadi customer baru. Form mengharuskan user untuk melengkapi data-data pada kontrol-kontrol textbox. Pada saat tombol daftar ditekan, sistem akan melakukan pengecekan ke basis data. Jika data yang di inputkan belum ada di basis data maka sistem akan menyimpan data baru tersebut, sebaliknya sistem akan meminta user untuk menginputkan data yang berbeda. Sistem akan akan melakukan redirect ke form edit profil customer apabila registrasi telah berhasil.

4.3 Use Case : Pengubahan Profil Customer

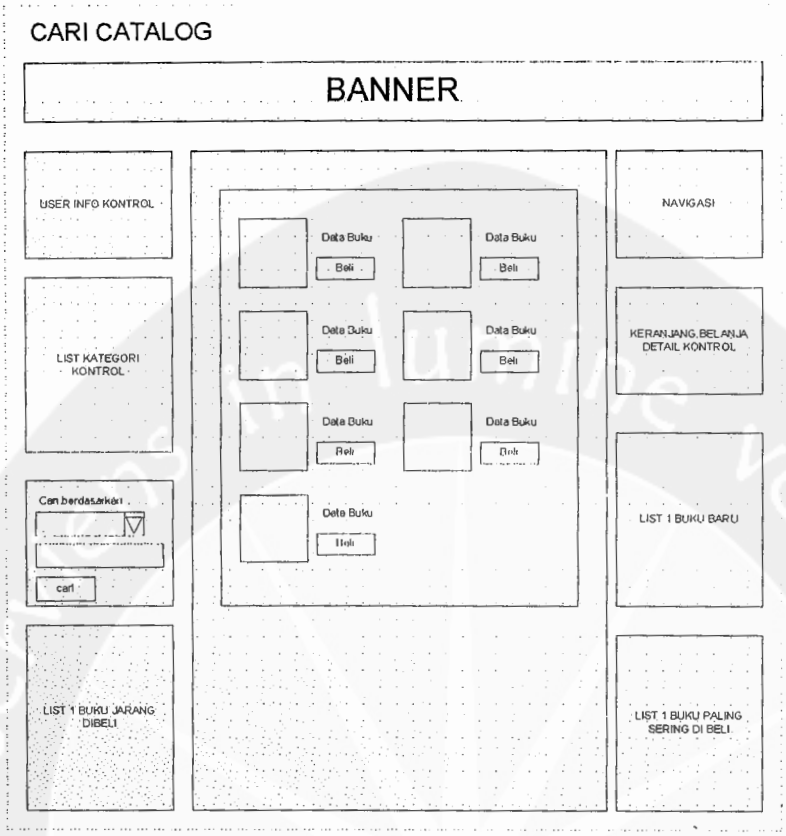


Gambar 4.3 Rancangan Antarmuka Form Pengubahan Profil Customer

Deskripsi

Antarmuka untuk mengubah data profil. Antarmuka ini berlaku bagi user yang mempunyai role sebagai customer. Form menyediakan beberapa kontrol textbox yang berhubungan dengan data profil customer, secara default kontrol berisi *null* data. User terdaftar dapat melakukan perubahan pada kontrol-kontrol tersebut. Pada saat tombol edit ditekan, sistem akan menyimpan data perubahan ke basis data. Data perubahan akan ditampilkan kembali melalui antarmuka ini.

4.4 Use Case : Pencarian Catalog



Gambar 4.4 Rancangan Antarmuka Form Pencarian Catalog

Deskripsi

Antarmuka untuk mencari data buku. User memilih terlebih dahulu key pencarian buku berdasarkan judul, penulis atau penerbit kemudian memasukan key pencarian. Penjelasan event yang terjadi yaitu: **cariButton_click,** sistem akan melakukan pencarian buku berdasarkan key yang telah diinputkan. Output pencarian ditampilkan dalam bentuk list buku pada container data.

Event :

Urutan aksi yang terjadi:

1. User memilih kategori pencarian pada control search
2. User memasukan buku yang akan dicari pada textbox.
3. User menekan tombol cari.
4. Sistem melakukan pencarian terhadap buku. Statement sql yang digunakan:

```
IF @PARAM1='JUDUL'
        SELECT
        ISBN, JUDUL,DESKRIPSI, PENULIS, PENERBIT, HARGA,
        DISKON,IMAGE , STATUSBARU, ID_KATEGORI
        FROM BUKU
        WHERE JUDUL LIKE '%'+@PARAM2+'%'
```

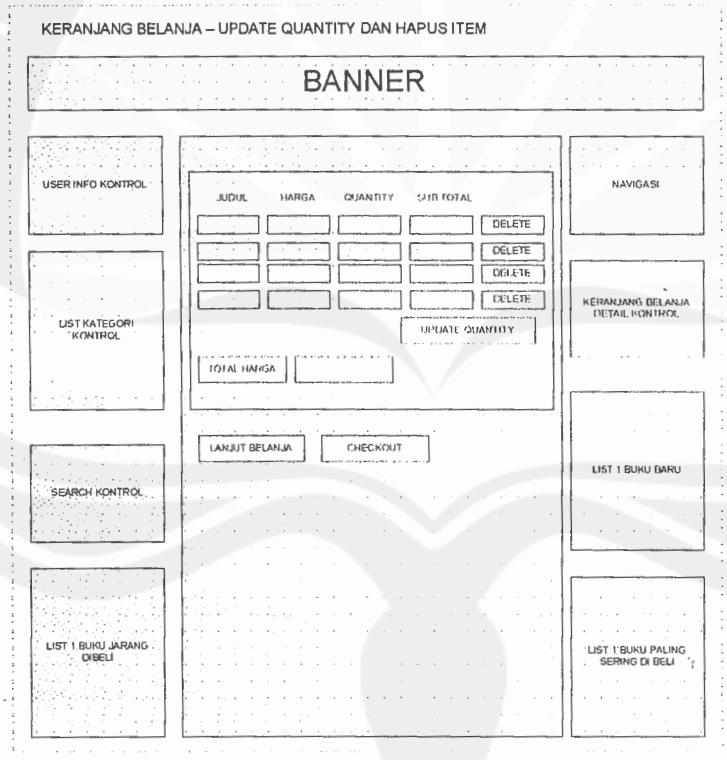
```
ELSE IF @PARAM1='PENERBIT'
        SELECT
        ISBN, JUDUL,DESKRIPSI, PENULIS, PENERBIT, HARGA,
        DISKON,IMAGE , STATUSBARU, ID_KATEGORI
        FROM BUKU
        WHERE PENERBIT LIKE '%'+@PARAM2+'%'

ELSE IF @PARAM 1='PENULIS'
        SELECT
        ISBN, JUDUL,DESKRIPSI, PENULIS, PENERBIT, HARGA,
        DISKON,IMAGE , STATUSBARU, ID_KATEGORI
        FROM BUKU
        WHERE PENULIS LIKE '%'+@PARAM2+'%'
```

Dengan [Param1] adalah key pencarian dan [Param2] adalah buku yang dicari.

5. Sistem akan menampilkan buku pada kontainer list buku

4.5 Use Case : Pengisian Keranjang Belanja

4.5.1 Form Edit Quantity dan Hapus Item Buku dari Keranjang Belanja



Gambar 4.5 Rancangan Antarmuka Form Keranjang Belanja – Update Quantity dan Hapus Item Buku dari Keranjang Belanja

Deskripsi

Antarmuka untuk mengedit quantity buku dan menghapus item buku dalam keranjang belanja. Form menampilkan data grid keranjang belanja. User

dapat melakukan perubahan terhadap data keranjang. Penjelasan event yang terjadi yaitu:

1. **updateButton_click,** sistem akan menyimpan data perubahan quantity di basis data. Data perubahan akan ditampilkan kembali melalui antarmuka ini.
2. **grid_RowDeleting,** sistem akan menghapus item buku pada keranjang belanja dari basis data. Data perubahan akan ditampilkan kembali melalui antarmuka ini.
3. **lanjutBelanja_click,** sistem akan melakukan redirect ke page terakhir dimana user melakukan proses tambah item ke keranjang belanja.
4. **checkout_click,** sistem akan melakukan redirect ke page checkout.

Event :

a) Urutan aksi yang terjadi pada update keranjang belanja:

1. Sistem menampilkan detail belanja.
2. User melakukan pengubahan quantity buku.
3. User menekan tombol update.
4. Sistem melakukan update jumlah quantity buku. Statement sql yang digunakan:

```
IF @PARAM1 <= 0
        EXEC DELETEKERANJANGBELANJA @IDKERANJANG, @ISBN
ELSE
        UPDATE DETAIL_KERANJANG_BELANJA
        SET QUANTITY = @PARAM1
        WHERE ISBN = @PARAM2 AND ID_KERANJANG_BELANJA = @PARAM3

        UPDATE KERANJANG_BELANJA
        SET TANGGAL_BELANJA = GETDATE()
        WHERE ID_KERANJANG_BELANJA = @PARAM3

        UPDATE STOK
        SET QUANTITY = QUANTITY - @SELISIH
        WHERE ISBN = @PARAM2
```

   [Param1] adalah quantity,[Param2] adalah ISBN dan [Param3] adalah Id keranjang belanja. jika Param1 lebih kecil atau sama dengan 0 maka dilakukan fungsi menghapus keranjang belanja. Sebaliknya akan dilakukan update detail keranjang belanja berdasarkan Param2 dan Param3, update keranjang belanja berdasarkan Param3, dan update stok berdasarkan Param2.

5. Sistem akan menampilkan kembali hasil update pada list keranjang belanja

b) Urutan aksi yang terjadi pada hapus keranjang belanja:

1. Sistem menampilkan detail belanja.
2. User menekan tombol hapus.
3. Sistem melakukan hapus terhadap keranjang belanja. Statement sql yang digunakan:

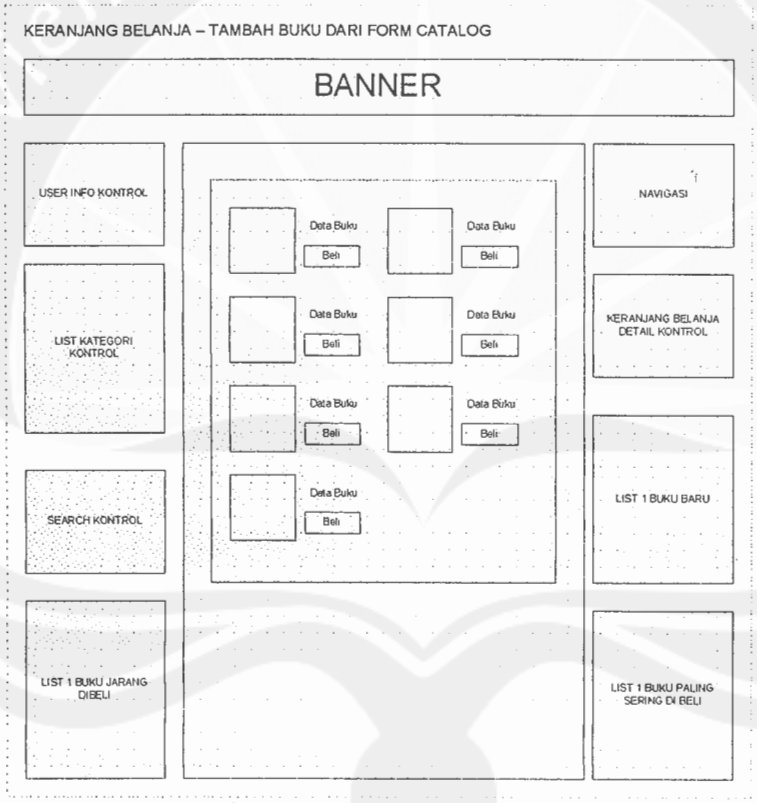
```
DELETE FROM DETAIL_KERANJANG_BELANJA
```

```
WHERE ID_KERANJANG_BELANJA = @PARAM1 AND ISBN = @PARAM2

IF NOT EXISTS ( SELECT ID_KERANJANG_BELANJA FROM
                DETAIL_KERANJANG_BELANJA
                WHERE ID_KERANJANG_BELANJA = @PARAM1)
    DELETE FROM KERANJANG_BELANJA
    WHERE ID_KERANJANG_BELANJA = @PARAM1
```

[Param1] adalah Id keranjang belanja dan [Param2] adalah ISBN. Proses yang dilakukan yaitu menghapus detail keranjang belanja terlebih dahulu dilanjutkan dengan menghapus keranjang belanja secara keseluruhan

4. Sistem akan menampilkan kembali hasil update pada list keranjang belanja.

4.5.2 Form Tambah Item Buku dari Form Catalog



Gambar 4.6 Rancangan Antarmuka Form Keranjang Belanja – Tambah Item Buku dari Form Catalog

Deskripsi

Antarmuka untuk memasukan item buku ke keranjang belanja. Form menampilkan data list buku kepada user. User dapat melakukan pembelian item buku melalui form ini. Penjelasan event yang terjadi yaitu: **list_ItemCommand**, Bila user menekan button beli pada data list maka sistem akan menambahkan buku pada keranjang belanja.

Event :

Urutan aksi yang terjadi:

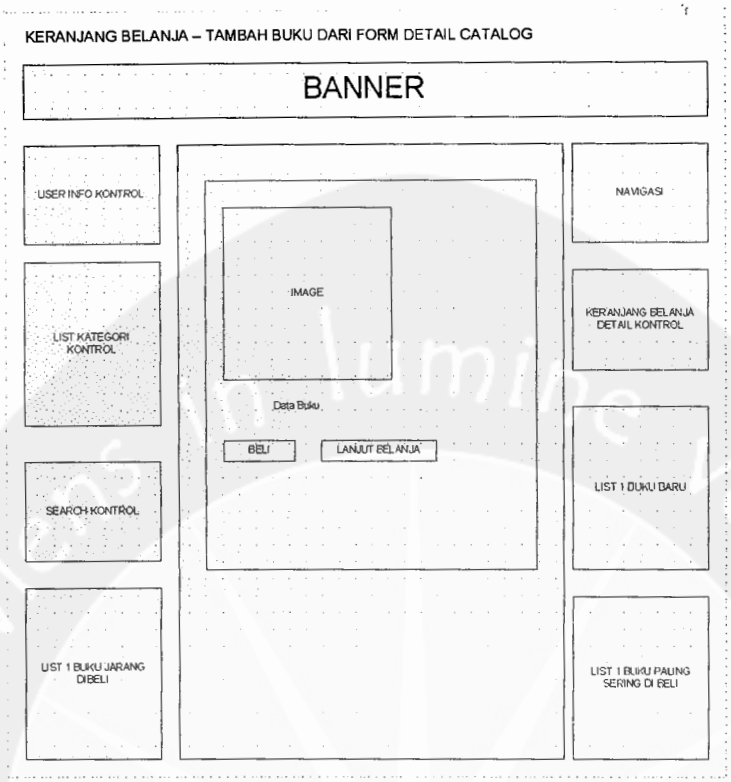
1. User memilih buku yang akan dibeli
2. User menekan tombol beli.
3. Sistem melakukan tambah buku ke keranjang belanja. Statement sql yang digunakan:

```
IF EXISTS(    SELECT ID_KERANJANG_BELANJA
              FROM DETAIL_KERANJANG_BELANJA
              WHERE ISBN = @PARAM2 AND ID_KERANJANG_BELANJA =
    @PARAM1)
    UPDATE DETAIL_KERANJANG_BELANJA
    SET QUANTITY = QUANTITY + 1
    WHERE ISBN = @PARAM2 AND ID_KERANJANG_BELANJA = @PARAM1

ELSE
IF EXISTS (SELECT JUDUL FROM BUKU WHERE ISBN=@PARAM2)

    INSERT INTO KERANJANG_BELANJA (ID_KERANJANG_BELANJA,
    TANGGAL_BELANJA,STATUS)
    VALUES (@PARAM1, GETDATE(),'ACTIVE')

    INSERT INTO DETAIL_KERANJANG_BELANJA (ID_KERANJANG_BELANJA,
    ISBN, QUANTITY)
    VALUES (@PARAM1, @PARAM2, 1)
```

[Param1] adalah Id keranjang belanja dan [Param2] adalah ISBN. Jika id keranjang belanja sudah tersimpan di basis data maka dilakukan update terhadap quantity buku berdasarkan Param1. Sebaliknya dilakukan insert data keranjang baru ke basis data.

4. Sistem akan menampilkan hasil update pada list keranjang belanja.

4.5.3 Form Tambah Item Buku dari Form Detail Catalog



Gambar 4.7 Rancangan Antarmuka Form Keranjang Belanja – Tambah Item Buku dari Form Detail Catalog

Deskripsi

Antarmuka untuk memasukan item buku ke keranjang belanja. Form menampilkan detail data buku kepada user. User juga dapat melakukan pembelian item buku melalui form ini. Penjelasan event yang terjadi yaitu:

1. **beliButton_click**, sistem akan menambahkan buku pada keranjang belanja.
2. **lanjutBelanja_click,** sistem akan melakukan redirect ke page terakhir dimana user melakukan proses tambah item ke keranjang belanja.

Event :

Urutan aksi yang terjadi:

1. User memilih buku yang akan dibeli
2. User menekan tombol beli.
3. Sistem melakukan tambah buku ke keranjang belanja. Statement sql yang digunakan:

```
IF EXISTS(    SELECT ID_KERANJANG_BELANJA
              FROM DETAIL_KERANJANG_BELANJA
              WHERE ISBN = @PARAM2 AND ID_KERANJANG_BELANJA =
@PARAM1)
UPDATE DETAIL_KERANJANG_BELANJA
SET QUANTITY = QUANTITY + 1
WHERE ISBN = @PARAM2 AND ID_KERANJANG_BELANJA = @PARAM1

ELSE
```

```
IF EXISTS (SELECT JUDUL FROM BUKU WHERE ISBN=@PARAM2)

    INSERT INTO KERANJANG_BELANJA (ID_KERANJANG_BELANJA,
    TANGGAL_BELANJA,STATUS)
    VALUES (@PARAM1, GETDATE(),'ACTIVE')

    INSERT INTO DETAIL_KERANJANG_BELANJA (ID_KERANJANG_BELANJA,
    ISBN, QUANTITY)
    VALUES (@PARAM1, @PARAM2, 1)
```

[Param1] adalah Id keranjang belanja dan [Param2] adalah ISBN. Jika id keranjang belanja sudah tersimpan di basis data maka dilakukan update terhadap quantity buku berdasarkan Param1. Sebaliknya dilakukan insert data keranjang baru ke basis data.

4. Sistem akan menampilkan hasil update pada list keranjang belanja.

4.6 Use Case : Order

Gambar 4.8 Rancangan Form Antarmuka Form Order

Deskripsi

Antarmuka untuk melakukan proses order. Form menampilkan detail keranjang belanja kepada user. Penjelasan event yang terjadi yaitu: **OrderButton_click**, pada saat user menekan tombol order, sistem menyimpan detail keranjang belanja tersebut menjadi suatu order kemudian sistem akan mengaktifkan proses order.

Event :

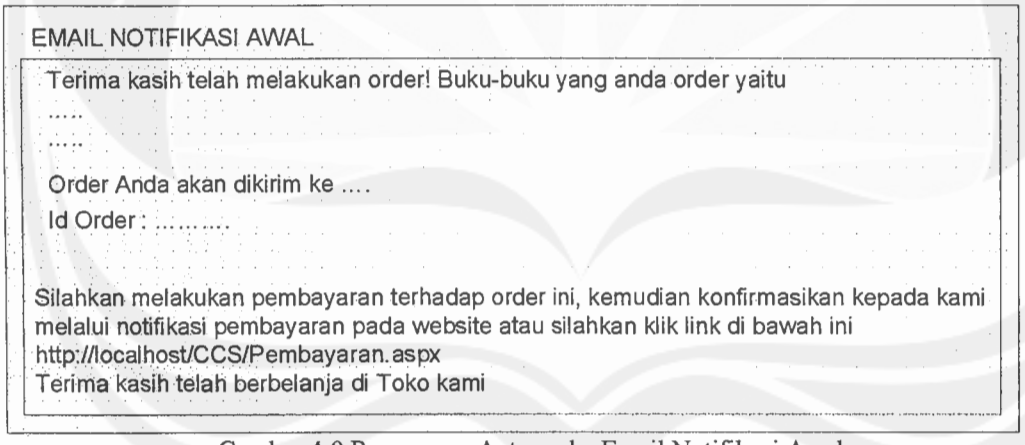
Urutan aksi yang terjadi:

1. Sistem menampilkan list keranjang belanja
2. User menekan tombol order.
3. Sistem melakukan create order baru. Statement sql yang digunakan:

```
INSERT INTO ORDERS (ID_CUSTOMER) VALUES (@IDCUSTOMER)

INSERT INTO DETAILORDER
(
    ID_ORDER, ISBN, QUANTITY
)
SELECT
    @IDORDER, BUKU.ISBN,
    DETAIL_KERANJANG_BELANJA.QUANTITY
FROM BUKU JOIN DETAIL_KERANJANG_BELANJA
ON BUKU.ISBN = DETAIL_KERANJANG_BELANJA.ISBN
WHERE DETAIL_KERANJANG_BELANJA.ID_KERANJANG_BELANJA =
@IDKERANJANG
```

[Param1] adalah Id keranjang. Proses yang terjadi yaitu menciptakan order baru dari list keranjang belanja.

4.7 Use Case : Email Notifikasi



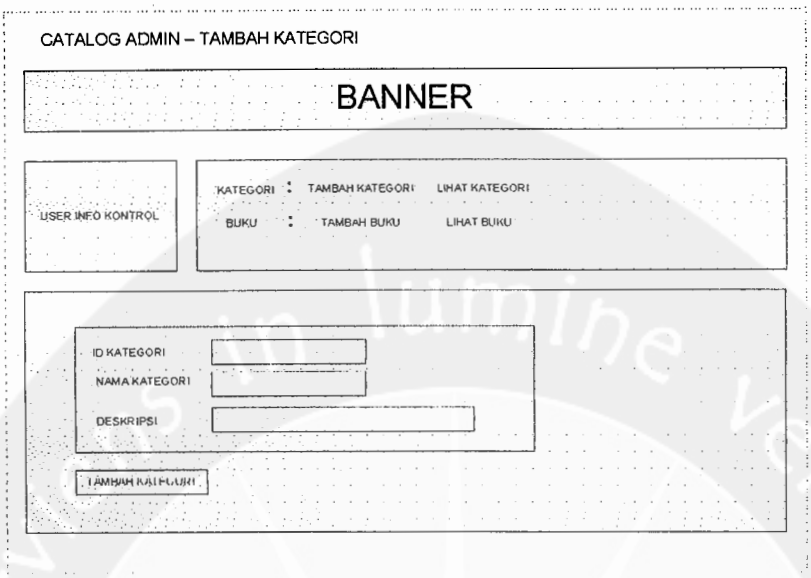
Gambar 4.9 Rancangan Antarmuka Email Notifikasi Awal

Deskripsi

Antarmuka email notifikasi awal kepada customer. Antarmuka ini merupakan bagian dari proses order yang disebabkan user atau customer telah melakukan proses order. Antarmuka ini berisi penjelasan mengenai order yang bersangkutan.

4.8 Use Case : Pengelolaan Catalog

4.8.1 Form Tambah Kategori



Gambar 4.10 Rancangan Antarmuka Form Pengelolaan Catalog – Tambah Kategori

Deskripsi

Antarmuka untuk menambah data kategori baru. Form menampilkan kontrol-kontrol textbox kepada administrator untuk menambah kategori baru. Administrator dapat menginputkan data kategori baru melalui antarmuka ini. Penjelasan event yang terjadi yaitu: **addKategori_click**, sistem akan melakukan pengecekan ke basis data. Jika data yang diinputkan belum ada di basis data maka sistem akan menyimpan data baru tersebut, sebaliknya sistem akan meminta user untuk menginputkan data yang berbeda.

Event :

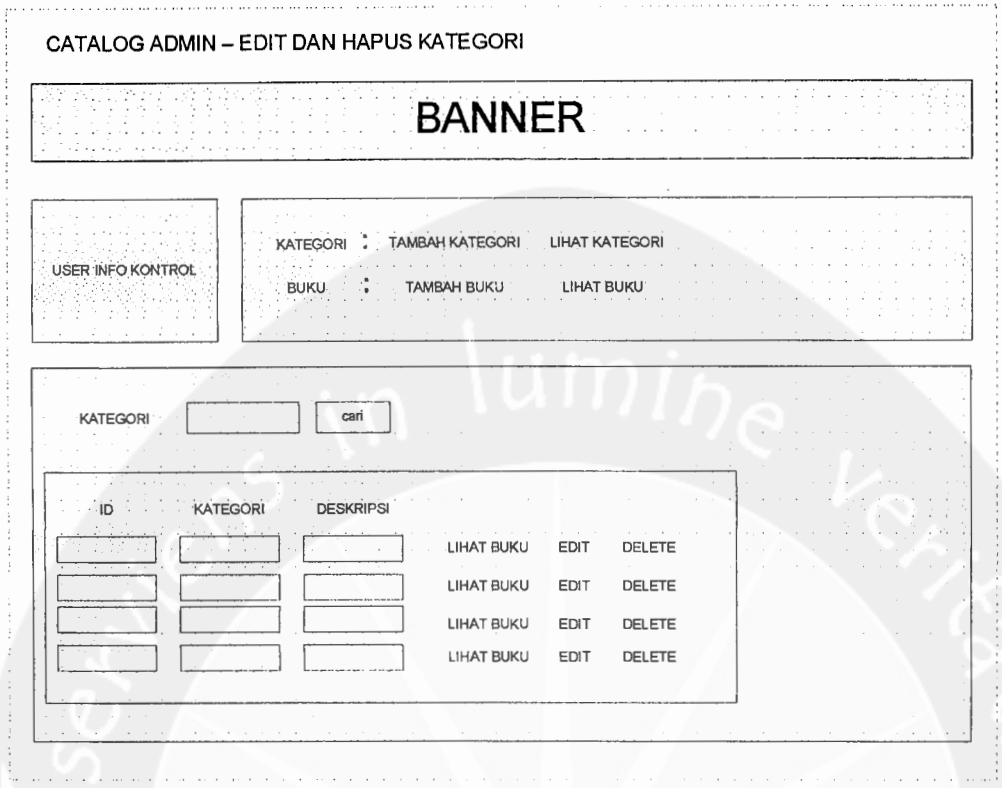
Urutan aksi yang terjadi:

1. Sistem menampilkan kontrol-kontrol textbox untuk isi data
2. User memasukan data kategori yang ingin ditambahkan.
3. User menekan tombol tambahh kategori
4. Sistem menyimpan kategori baru ke basis data. Statement sql yang digunakan:

```
INSERT INTO KATEGORI (ID_KATEGORI, NAMA_KATEGORI,DESKRIPSI)
VALUES (@PARAM1, @PARAM2, @PARAM3)
```

[Param1] adalah I kategori, [Param2] adalah Nama kategori dan [Param3] adalah deskripsi. Proses yang terjadi yaitu insert kategori baru ke basis data dengan parameter Param1, Param2 dan Param3.

4.8.2 Form Edit dan Hapus Kategori



Gambar 4.11 Rancangan Antarmuka Form Pengelolaan Catalog dmin – Edit dan Hapus Kategori

Deskripsi

Antarmuka untuk melakukan edit terhadap data kategori dan menghapus data kategori. Antarmuka menampilkan list data kategori kepada administrator. Administrator juga dapat memilih langsung kategori dengan menginputkan key untuk melakukan pencarian data kategori. Penjelasan event yang terjadi yaitu:

5 **grid_RowDeleting,** sistem menghapus kategori yang dari basis data
6 **grid_RowUpdating,** sistem mengupdate data kategori dan menyimpannya kembali ke basis data.
7 **Lihat buku textButton**, system akan meredirect ke page yang menampilkan list buku.

Event :

a) Urutan aksi yang terjadi pada edit kategori:
   1. Sistem menampilkan list kategori.
   2. User menekan tombol edit.
   3. Sistem mengaktifkan form edit.
   4. User mengubah data kategori.
   5. User menkan tombol update.
   6. Sistem melakukan update. Statement sql yang digunakan:

```
UPDATE KATEGORI
SET NAMA_KATEGORI=@PARAM1,
```

```
    DESKRIPSI=@PARAM2
WHERE ID_KATEGORI=@PARAM3
```

[Param1] adalah nama kategori,[Param2] adalah deskripsi dan [Param3] adalah Id kategori.

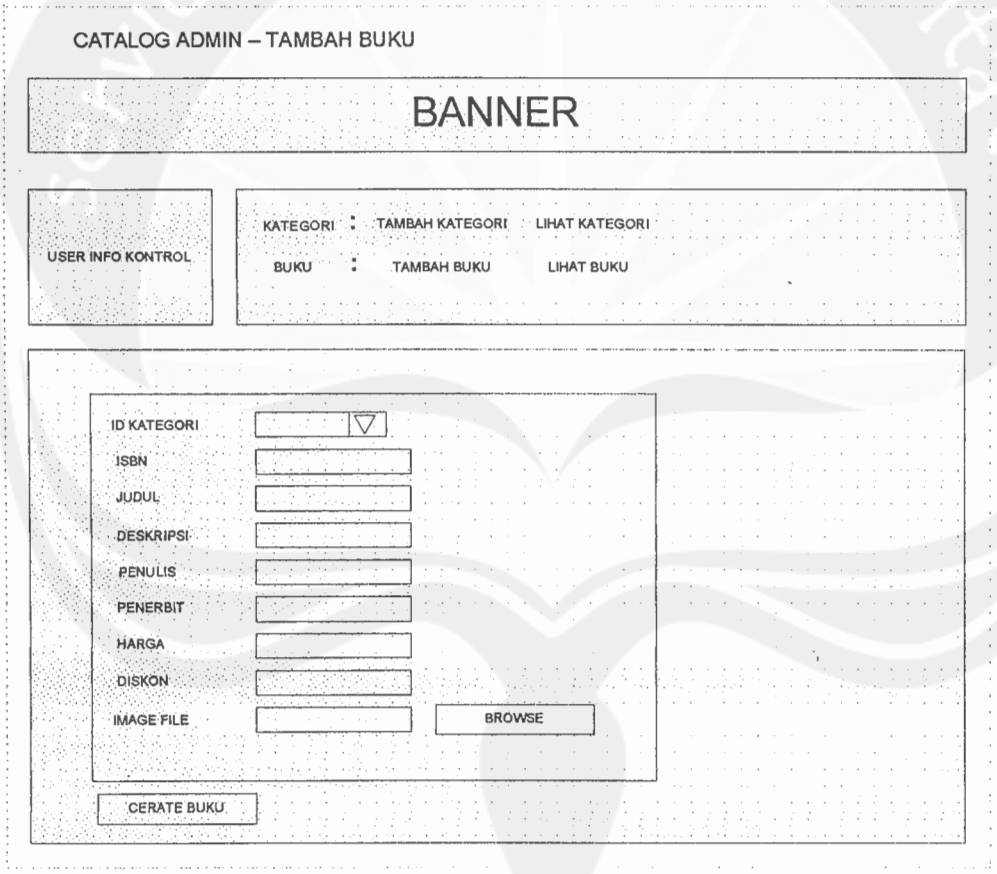
b) Urutan aksi yang terjadi pada hapus kategori:

1. Sistem menampilkan list kategori.
2. User menekan tombol hapus.
3. Sistem melakukan hapus terhadap kategori. Statement sql yang digunakan:

```
DELETE FROM KATEGORI
WHERE ID_KATEGORI=@PARAM1
```

[Param1] adalah Id kategori.

8.8.3 Form Tambah Buku



Gambar 4.12 Rancangan Antarmuka Form Pengelolaan Catalog – Tambah Buku

Deskripsi

Antarmuka untuk menambah data buku baru. Form menampilkan kontrol-kontrol textbox kepada administrator untuk menambah buku baru. Administrator

dapat menginputkan data buku baru melalui antarmuka ini. Penjelasan event yang terjadi yaitu: **createBuku_click**, sistem akan melakukan pengecekan ke basis data. Jika data yang diinputkan belum ada di basis data maka sistem akan menyimpan data baru tersebut, sebaliknya sistem akan meminta user untuk menginputkan data yang berbeda.

Event :

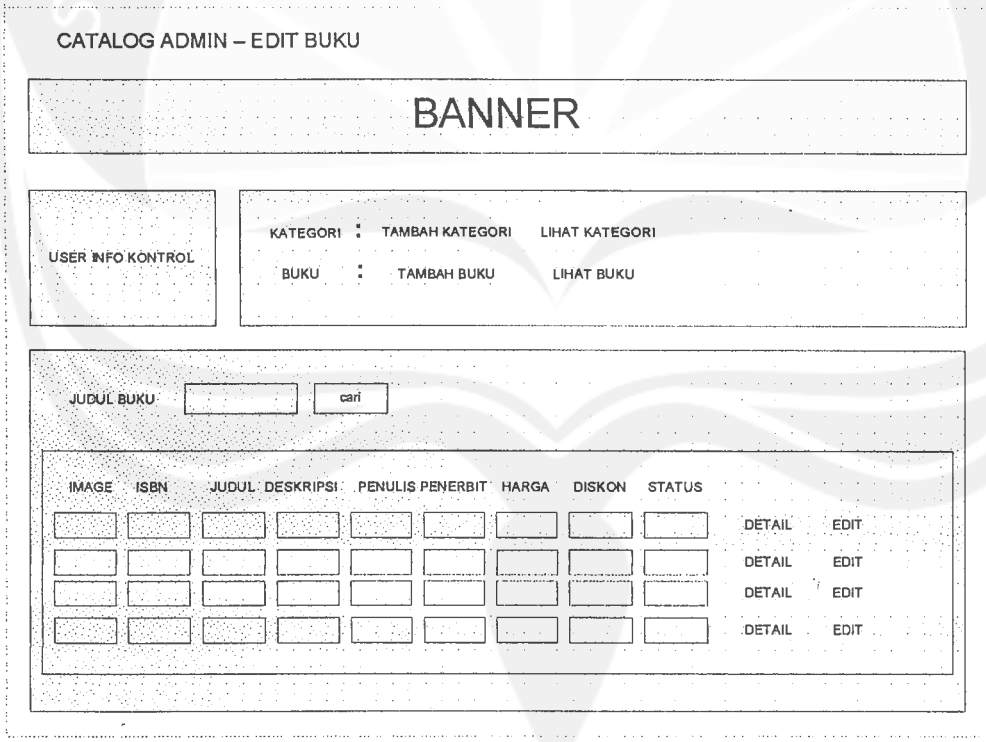
Urutan aksi yang terjadi:

1. Sistem menampilkan kontrol-kontrol textbox untuk isi data
2. User memasukan data buku yang ingin ditambahkan.
3. User menekan tombol tambah buku
4. Sistem menyimpan buku baru ke basis data. Statement sql yang digunakan:

```
INSERT INTO BUKU
VALUE(@PARAM1 ,@PARAM2 ,@PARAM3,@PARAM4
,@PARAM5,@PARAM6,@PARAM7 ,@PARAM8 ,@PARAM9)
```

[Param1] adalah Isbn, [Param2] adalah Judul kategori, [Param3] adalah deskripsi, [Param4] adalah penulis, [Param5] adalah penerbit, [Para6] adalah harga, [Param7] adalah diskon, [Param8] adalah image, [Param9] adalah Id kategori.

8.8.4 Form Edit Buku



Gambar 4.13 Rancangan Antarmuka Form Pengelolaan Catalog– Edit Buku

Deskripsi

Antarmuka untuk melakukan edit terhadap data buku. Antarmuka menampilkan list data buku kepada administrator. Administrator juga dapat memilih langsung kategori dengan menginputkan key untuk melakukan pencarian data buku. Penjelasan event yang terjadi yaitu:

1 **grid_RowUpdating,** sistem mengupdate data kategori dan menyimpannya kembali ke basis data.
2 **Detail textButton**, sistem akan meredirect ke page yang menampilkan list detail buku.

Event :

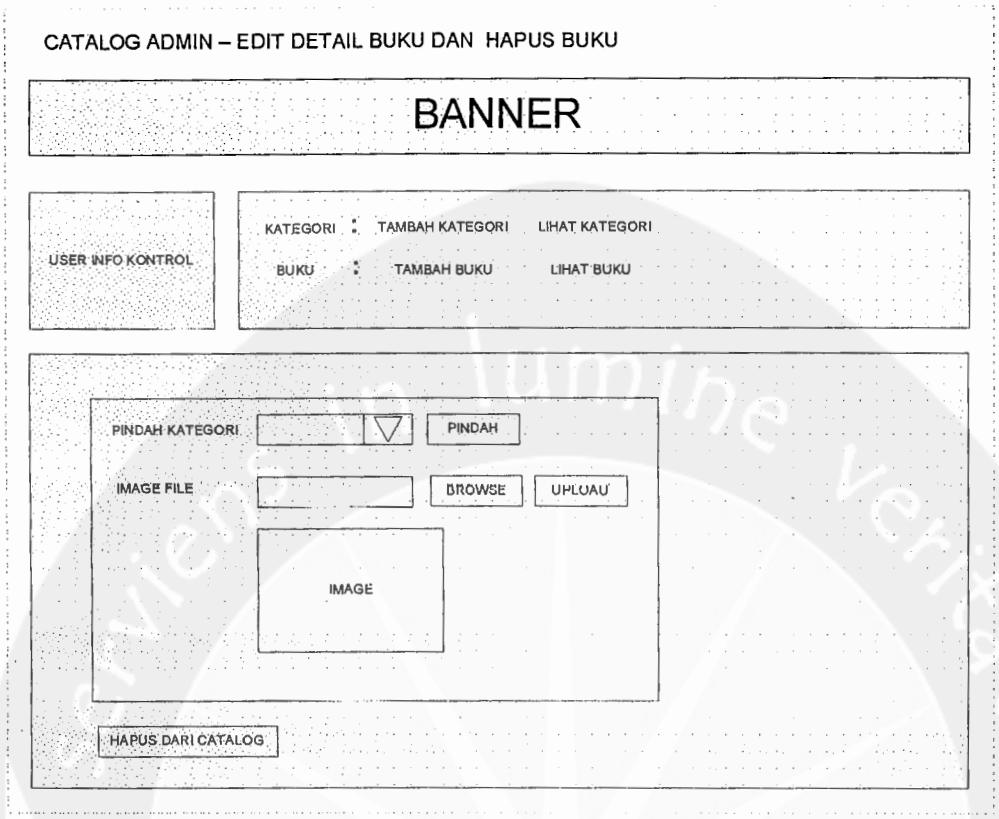
Urutan aksi yang terjadi pada edit buku:

1. Sistem menampilkan list buku.
2. User menekan tombol edit.
3. Sistem mengaktifkan form edit.
4. User mengubah data buku.
5. User menekan tombol update.
6. Sistem melakukan update. Statement sql yang digunakan:

```
UPDATE BUKU
SET JUDUL=@PARAM2,
      DESKRIPSI=@PARAM3,
      PENULIS=@PARAM4,
      PENERBIT=@PARAM5,
      HARGA=@PARAM6,
      DISKON=@PARAM7,
      STATUSBARU=@PARAM8,
      IMAGE=@PARAM9
WHERE ISBN=@PARAM1
```

[Param1] adalah Isbn, [Param2] adalah Judul kategori, [Param3] adalah deskripsi, [Param4] adalah penulis, [Param5] adalah penerbit, [Para6] adalah harga, [Param7] adalah diskon, [Param8] adalah status baru, [Param9] adalah image.

4.8.5 Form Edit Detail Buku dan Hapus Buku



Gambar 4.14 Rancangan Antarmuka Pengelolaan Catalog– Edit Detail Buku dan Hapus Buku

Deskripsi

Antarmuka untuk melakukan edit terhadap detail data buku. Pada antamuka ini kategori buku dapat diubah dan dapat memperbaharui image file buku. Penjelasan event yang terjadi yaitu:

1 **grid_RowDeleting,** sistem menghapus data buku dari basis data.
2 **pindahBt_Click,** sistem mengubah kategori buku dan menyimpannya ke basis data.
3 **uploadButton_Click,** fungsi untuk mengupload image buku.

Event :

a) Urutan aksi yang terjadi pada pindah kategori:
   1. User memilih kategori yang akan menjadi kategori kategori baru buku.
   2. User menekan tombol pindah.
   3. Sistem melakukan pengubahan kategori. Statement sql yang digunakan:

```
UPDATE BUKU
SET ID_KATEGORI=@PARAM1  WHERE ISBN=@PARAM2
```

[Param1] adalah id kategori dan [Param2] adalah isbn.

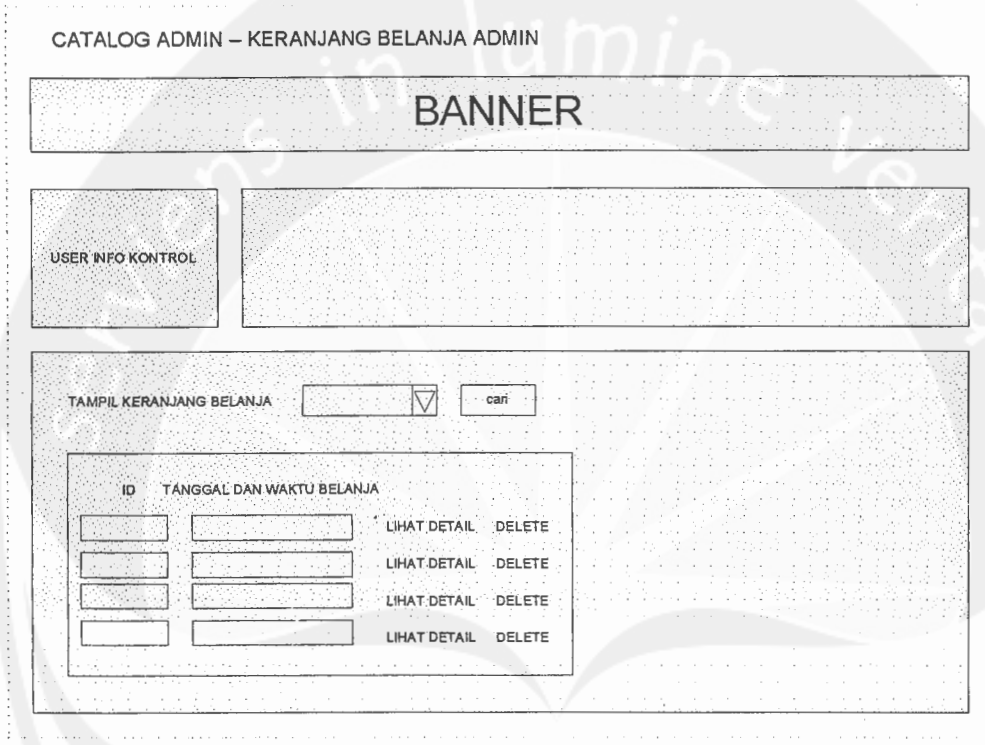
b) Urutan aksi yang terjadi pada hapus buku:

1. User menekan tombol hapus.
2. Sistem melakukan hapus terhadap kategori. Statement sql yang digunakan:

```
DELETE FROM BUKU
WHERE ISBN=@PARAM1
```

[Param1] adalah Isbn.

4.9 Use Case : Pengelolaan Keranjang Belanja



Gambar 4.15 Rancangan Antarmuka Form Pengelolaan Keranjang Belanja

Deskripsi

Antarmuka untuk melakukan reorganisasi keranjang belanja. Antarmuka menampilkan list data keranjang belanja kepada administrator. Administrator juga dapat memilih langsung kategori melakukan pencarian terhadap data keranjang belanja yang akan direorganisasi. Penjelasan event yang terjadi yaitu:

1 **grid_RowDeleting,** sistem menghapus data keranjang belanja dari basis data.
2 **Detail textButton**, sistem akan meredirect ke page yang menampilkan list detail keranjang belanja.

Event :

Urutan aksi yang terjadi pada hapus keranjang belanja:

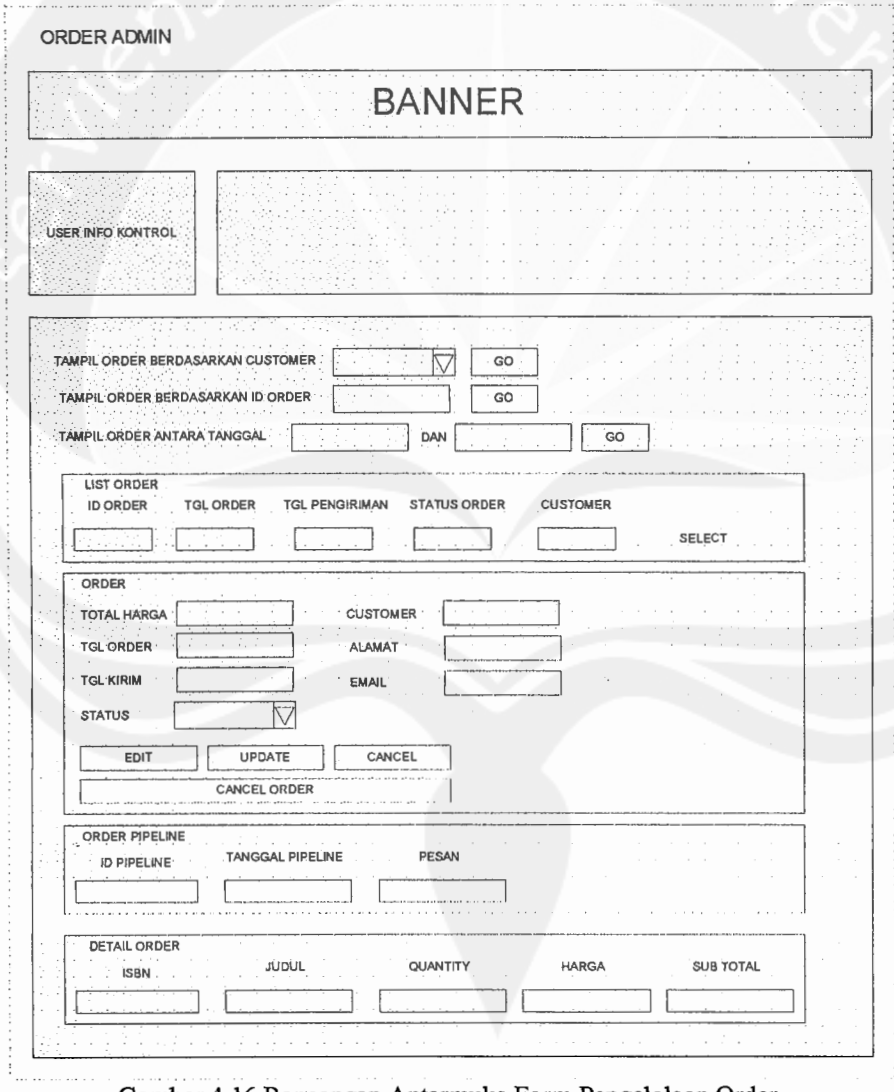
1. User memilih list keranjang belanja berdasarkan waktu.
2. User menekan tombol tampil.

3. User memilih keranjang belanja dalam list.
3. User menekan tombol hapus.
4. Sistem melakukan hapus terhadap data keranajang belanja. Statement sql yang digunakan:

```
DELETE FROM DETAIL_KERANJANG_BELANJA
WHERE ID_KERANJANG_BELANJA = @PARAM1

DELETE FROM KERANJANG_BELANJA
WHERE ID_KERANJANG_BELANJA = @PARAM1
```

[Param1] adalah Id keranjang belanja. Prosesnya yaitu menghapus detail keranjang belanja terlebih dahulu dilanjutkan dengan menghapus keranjang belanja keseluruhan

4.10 Use Case : Pengelolaan Order



Gambar 4.16 Rancangan Antarmuka Form Pengelolaan Order

Deskripsi

Antarmuka untuk pengelolahan order admin. Form menampilkan list order, list detail order dan list order pipeline. Administrator dapat melakukan pengelolahan order melalui form ini diantaranya yaitu edit data order dan meng-cancel order. Administrator memilih data order yang akan di ubah atau dilihat dengan memasukan key pencarian, pada saat button go ditekan maka sistem akan mencari data yang dimaksud ke basis data dan menampilkannya kembali melalui antarmuka ini. Penjelasan event yang terjadi yaitu:

1 **byIDGo_Click,** sistem mencari data order berdasarkan Id dari basis data.
2 **byDateGo_Click,** sistem mencari data order berdasarkan tanggal dari basis data.
3 **byCustomerGo_Click,** sistem mencari data order berdasarkan customer dari basis data.
4 **editButton_Click**, untuk mengaktifkan mode edit pada kontrol-kontrol textbox.
5 **updateButton_Click**, sistem akan mengupdate data order dan menyimpannya ke basis data.
6 **cancelButton_Click**, untuk menonaktifkan mode edit pada kontrol-kontrol textbox.
7 **cancelOrderButton_Click**, sistem akan mengubah status order pada order yang aktif menjadi cancel.

Event :

a) Urutan aksi yang terjadi pada edit order:
1. User memilih list order.
2. Sistem menampilkan list order
3. User memilih salah-satu order.
4. Sistem menampilkan detail order.
5. User menekan tombol edit.
6. Sistem mengaktifkan form edit.
7. User mengubah status order.
8. User menekan tombol update.
9. Sistem melakukan pengubahan status order. Statement sql yang digunakan:

```
UPDATE ORDERS
SET TANGGAL_ORDER=@PARAM1,
TANGGAL_PENGIRIMAN=@PARAM2,
STATUS_ORDER=@PARAM3
WHERE ID_ORDER = @PARAM4
```

[Param1] adalah tanggal order, [Param2] adalah tanggal pengiriman, [Param3] adalah status order dan [Param4] adalah Id order.

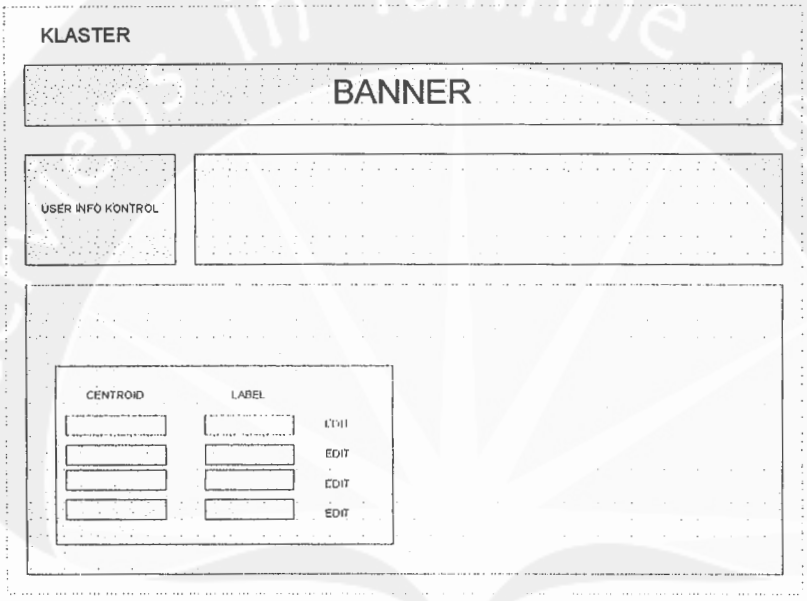
b) Urutan aksi yang terjadi pada cancel order:
1. User memilih list order.
2. Sistem menampilkan list order

3. User memilih salah-satu order.
4. Sistem menampilkan detail order.
5. User menekan tombol edit.
6. Sistem melakukan hapus terhadap kategori. Statement sql yang digunakan:

```
UPDATE ORDERS
SET STATUS_ORDER = @PARAM1
WHERE ID_ORDER = @PARAM2
```

[Param1] adalah status order dan [Param2] adalah id order.

4.11 Use Case : Pengelolaan Klaster

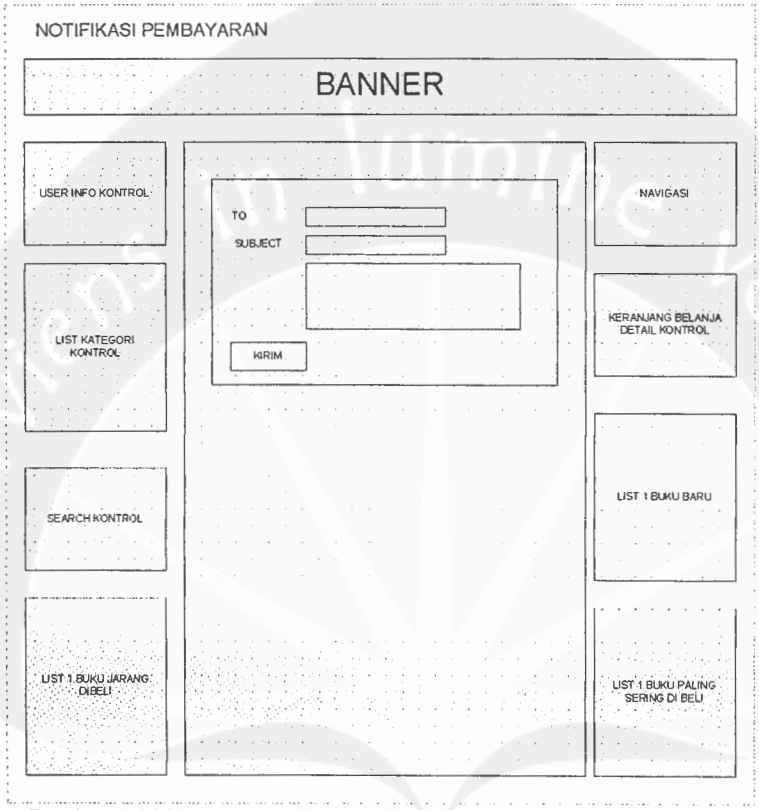


Gambar 4.17 Rancangan Antarmuka Form Pengelolaan Klaster

Deskripsi

Antarmuka untuk mengubah label klaster. Form menampilkan list data klaster. Administrator dapat melakukan pengubahan terhadap label klaster. Penjelasan event yang terjadi yaitu: **grid_RowUpdating**, sistem mengupdate label klaster dan menyimpannya kembali ke basis data.

4.12 Use Case : Notifikasi Pembayaran

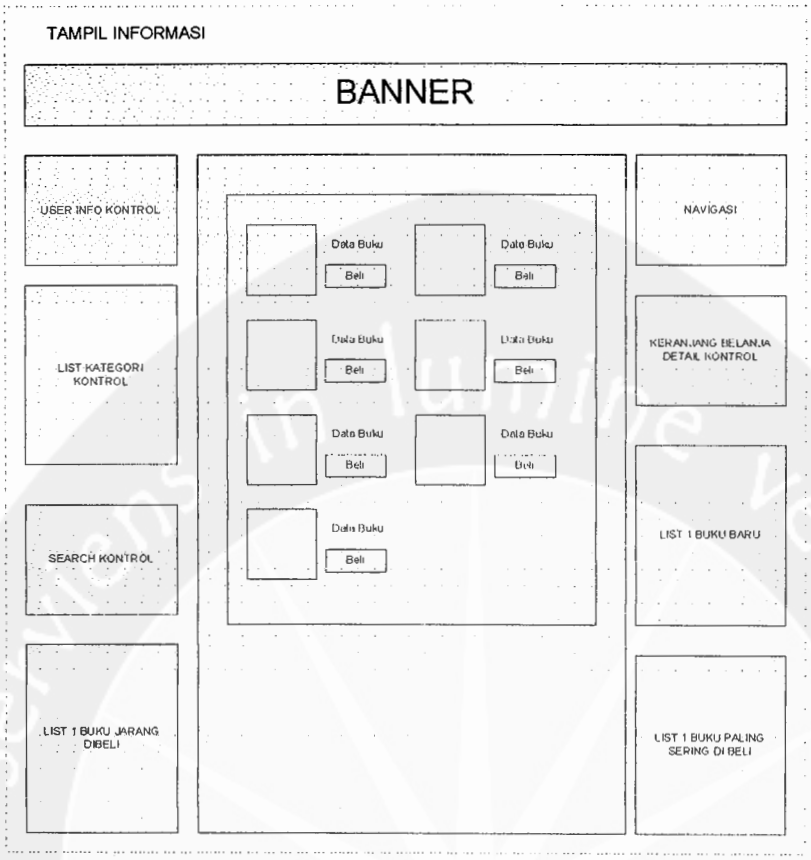


Gambar 4.18 Rancangan Antarmuka Form Notifikasi Pembayaran

Deskripsi

Antarmuka untuk mengirim email ke admin order. Form menampilkan antarmuka untuk mengirim email. Form ini digunakan oleh customer untuk memberitahu admin order bahwa pembayaran telah dilakukan. Customer diharuskan mengisi kontrol-kontrol textbox. Pada saat button kirim ditekan, sistem akan mengaktifkan orderProcessorMailer untuk mengirim email ke admin order tentang pemberitahuan pembayaran tersebut.

4.13 Use Case : Tampil Informasi

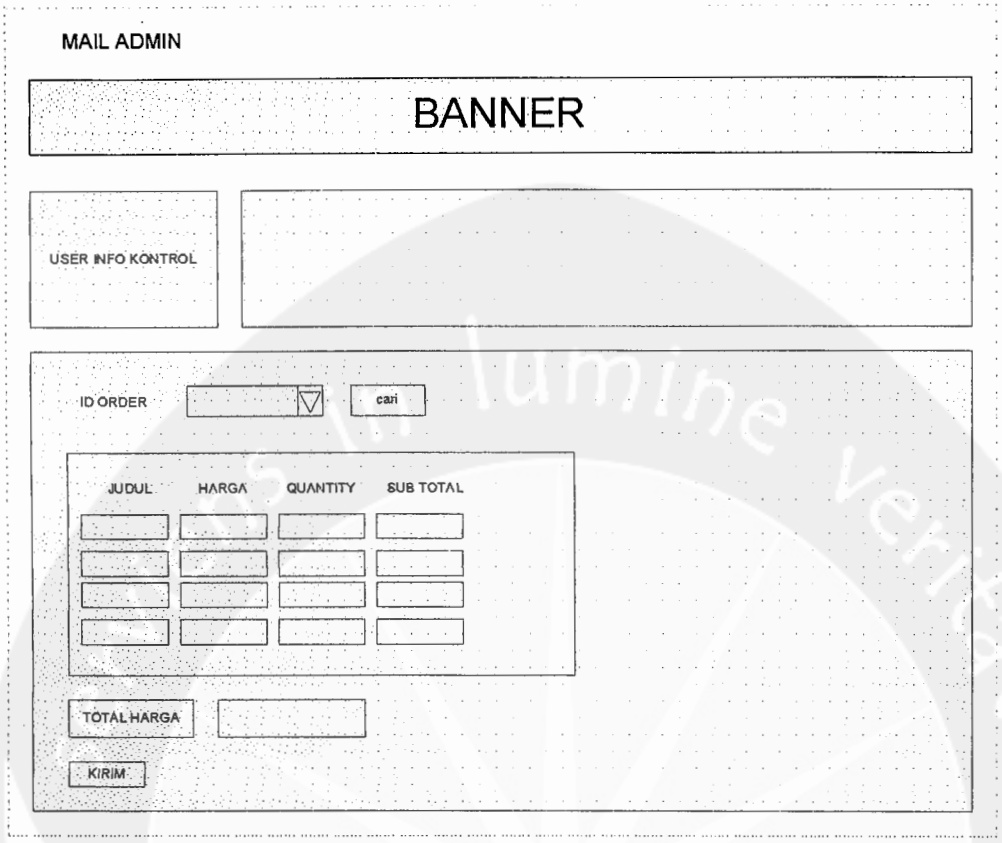


Gambar 4.19 Rancangan Antarmuka Form Tampil Informasi

Deskripsi

Antarmuka untuk menampilkan list catalog. Customer dapat memilih kategori dari list catalog yang akan ditampilkan dengan menclick kategori pada list kategori kontrol, output list catalog akan ditampilkan pada container data. Untuk melihat detail dari item pada catalog, customer dapat mengclicknya pada image item tersebut kemudian sistem akan meredirect ke page yang menampilkan detail item tersebut.

4.14 Use Case : Pengelolaan Email

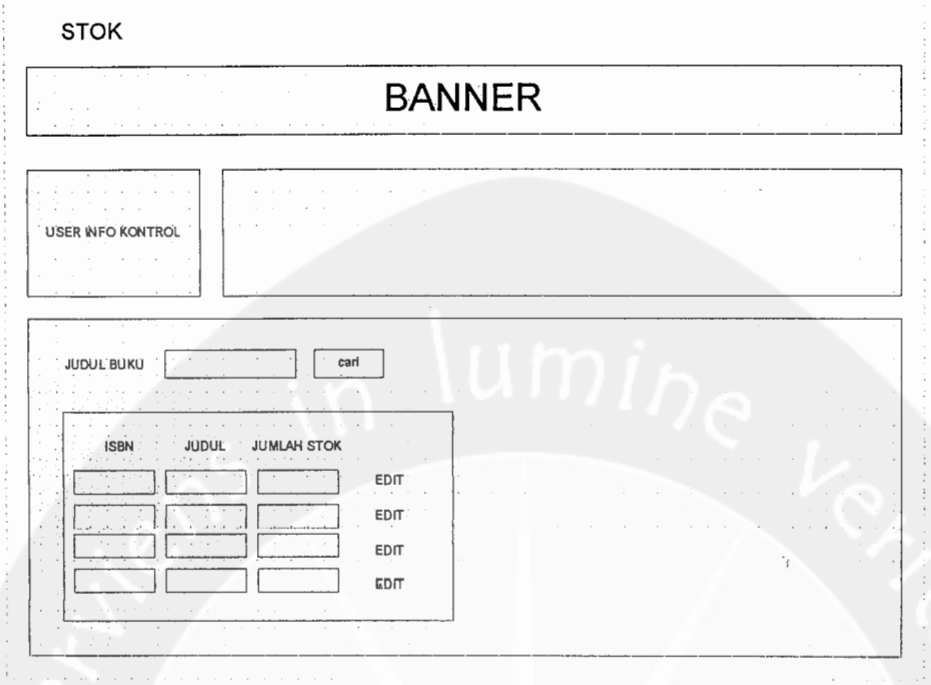


Gambar 4.20 Rancangan Antarmuka Form Pengelolaan Email

Deskripsi

Antarmuka untuk mengirim email notifikasi ke customer. Administrator memilih Id Order yang akan dikirim email kemudian menekan button cari, sistem menampilkan list detail order dari id order tersebut. Jika status id order tersebut belum complete maka akan tampil button kirim yang berfungsi untuk mengirim email notifikasi ke administrator. Email notifikasi dapat berupa email mengenai pembayaran telah diterima ataupun pengiriman barang tergantung dari status ordernya. Jika Administrator menekan button kirim maka email akan segera dikirim ke customer.

4.15 Use Case : Pengelolaan Stok



Gambar 4.21 Rancangan Antarmuka Form Pengelolaan Stok

Deskripsi

Antarmuka untuk mengubah jumlah stok. Form menampilkan list data stok. Administrator juga dapat memilih stok berdasarkan judul buku tertentu kemudian melakukan pengubahan terhadap jumlah stok. Penjelasan event yang terjadi yaitu: **grid_RowUpdating**, sistem mengupdate jumlah stok dan menyimpannya kembali ke basis data.

Event :

Urutan aksi yang terjadi:

1. Sistem menampilkan list stok buku.
2. User menekan tombol edit.
3. Sistem mengaktifkan form edit.
4. User mengubah stok buku buku.
5. User menekan tombol update.
6. Sistem melakukan update. Statement sql yang digunakan:

```
UPDATE STOK
SET QUANTITY=@QUANTITY WHERE ISBN=@ISBN
```

[Param1] adalah quantity dan [Param2] adalah isbn.