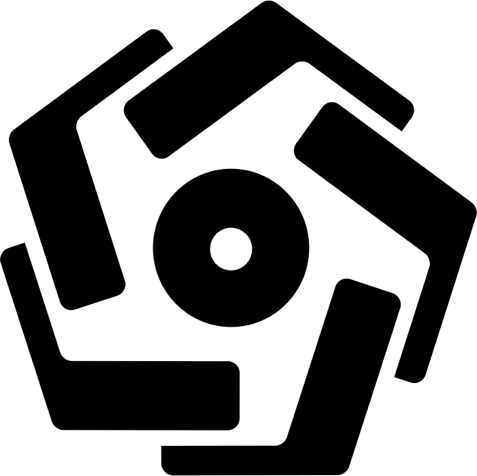
**PERANCANGAN SISTEM INFORMASI PENJUALAN ONLINE MELALUI WEBSITE DI TOKO SABLON KAOS NGAPAK**

**LAPORAN TUGAS RPL**

****

Disusun oleh:

**16-S1SI-06**

Herbarata Yuda Laksmana (16.12.9381)

Nur Muhammad Yasin (16.12.9377)

Muhammad Willy Iqbal H. (16.12.9376)

Adhy Syahputra (16.12.9361)

Kukuh Budiarto (16.12.9348)

**PRODI SISTEM INFORMASI**

**FAKULTAS ILMU KOMPUTER**

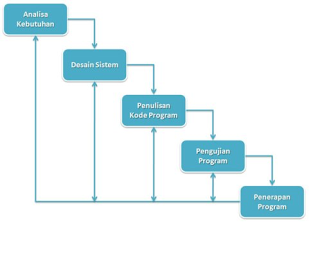
**UNIVERSITAS AMIKOM YOGYAKARTA**

**YOGYAKARTA**

**2019**

**3.5 *Software Proses Model***

*Software* proses model yang digunakan oleh Toko Sablon Kaos Ngapak adalah *waterfall*. Proses *waterfall* digunakan karena dalam pembuatan sistem informasi penjualan *online* berbasis *website* tidak akan mendapat pengembangan. Sehingga hanya memiliki fungsi penjualan.

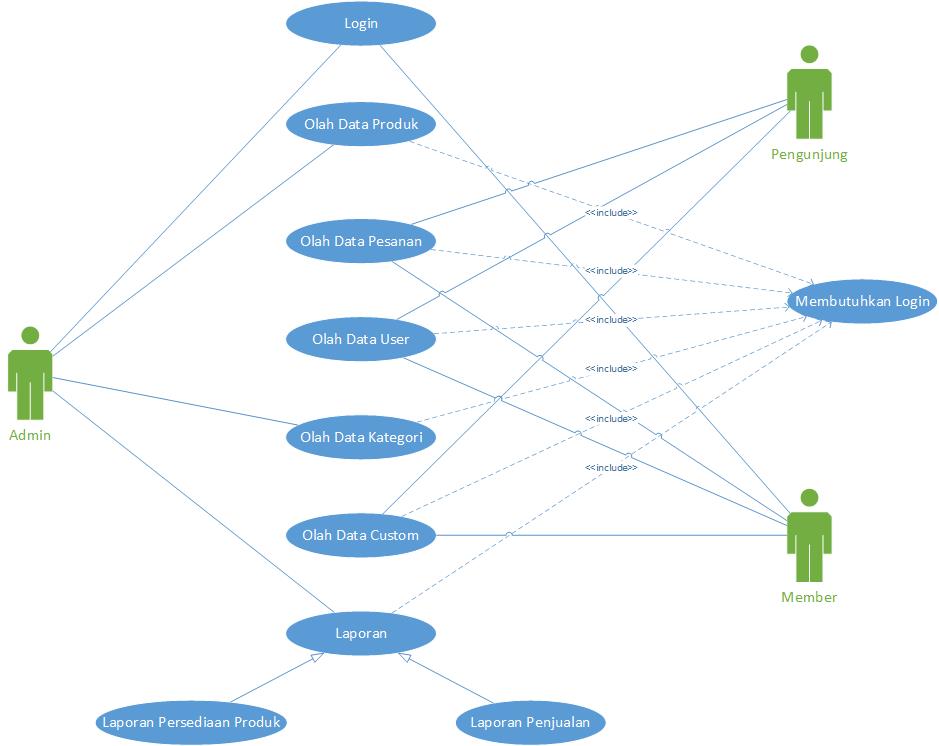


**Gambar 3.20 Proses Model *Waterfall***

**3.6 *Unified Modeling Language* (UML)**

**3.6.1 *Use Case* Diagram**

*Use case* diagramadalah alat komunikasi tingkat tinggi untuk mewakili persyaratan sistem. Diagram menunjukkan interaksi antara pengguna dan entitas eksternal lainnya dengan sistem yang sedang dikembangkan. Sistem dapat melakukan beberapa proses yaitu proses *login*, olah data produk, olah data pesanan, olah data *user*, olah data kategori, laporan penjualan, dan laporan persediaan produk. Terdapat tiga *user* yang dapat mengakses sistem yaitu admin, pengunjung, dan member. Berikut ini adalah gambar *use case* diagramToko Sablon Kaos Ngapak.



**Gambar 3.21 *Use Case* Diagram**

* Deskripsi *Use Case Login*

**Tabel 3.17 Deskripsi *Use Case Login***

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Login** |
| Aktor | : | Admin dan member |
| *Description* | : | *Use case* ini digunakan untuk *login* |
| *Pre Condition* | : | Form *login* |
| *Post Condition* | : | Halaman sesuai data *login* |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor mengisikan data *username* dan *password* |  | |
| 1. Klik tombol *Login* |  | |
|  | 1. Sistem akan melakukan pengecekan data *username* dan *password* | |
|  | 1. Jika ada yang belum diisi, maka sistem akan menampilkan *MESSAGE* “MAAF USERNAME ATAU PASSWORD ANDA SALAH” | |
|  | 1. Jika data yang diisikan telah sesuai, maka login akan berhasil | |

* Deskripsi *Use Case* Olah Data Produk

**Tabel 3.18 Deskripsi *Use Case* Olah Data Produk**

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Produk** |
| Aktor | : | Admin |
| *Description* | : | *Use case* ini digunakan untuk mengelola data produk, menambahkan, mengedit, dan menghapus |
| *Pre Condition* | : | Daftar produk ditampilkan |
| *Post Condition* | : | Mencetak laporan persediaan produk |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor akan mengisikan data produk yang akan ditambah |  | |
| 1. Klik tombol Tambah |  | |
|  | 1. Sistem akan melakukan pengecekan data produk yang diisikan apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka sistem akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data produk akan tersimpan di tabel produk | |
| Alternatif : *update* data produk |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih data yang akan dirubah dan melakukan perubahan data |  | |
| 1. Klik tombol UPDATE |  | |
|  | 1. Sistem akan melakukan pengecekan data yang diubah apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka system akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data produk akan tersimpan di tabel produk | |
| Alternatif : *delete* data produk |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih produk yang akan dihapus |  | |
| 1. Klik tombol HAPUS |  | |
|  | 1. Sistem akan menampilkan *message* “YAKIN DATA INI AKAN DIHAPUS ?” | |
| 1. Aktor klik button YA |  | |
|  | 1. Data yang dihapus akan hilang dari tabel produk | |

* Deskripsi *Use Case* Olah Data Kategori

**Tabel 3.19 Deskripsi *Use Case* Olah Data Kategori**

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Kategori** |
| Aktor | : | Admin |
| *Description* | : | *Use case* ini digunakan untuk mengelola data kategori, menambahkan, mengedit, dan menghapus |
| *Pre Condition* | : | Daftar kategori ditampilkan |
| *Post Condition* | : | Mencetak laporan persediaan kategori |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor akan mengisikan data kategori yang akan ditambah |  | |
| 1. Klik tombol Tambah |  | |
|  | 1. Sistem akan melakukan pengecekan data kategori yang diisikan apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka sistem akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data kategori akan tersimpan di tabel kategori | |
| Alternatif : *update* data kategori |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih data yang akan dirubah dan melakukan perubahan data |  | |
| 1. Klik tombol UPDATE |  | |
|  | 1. Sistem akan melakukan pengecekan data yang diubah apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka system akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data kategori akan tersimpan di tabel kategori | |
| Alternatif : *delete* data kategori |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih kategori yang akan dihapus |  | |
| 1. Klik tombol HAPUS |  | |
|  | 1. Sistem akan menampilkan *message* “YAKIN DATA INI AKAN DIHAPUS ?” | |
| 1. Aktor klik button YA |  | |
|  | 1. Data yang dihapus akan hilang dari tabel kategori | |

* Deskripsi *Use Case* Olah Data Pesanan

**Tabel 3.20 Deskripsi *Use Case* Olah Data Pesanan**

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Pesanan** |
| Aktor | : | Member dan pengunjung |
| *Description* | : | *Use case* ini digunakan untuk menjelaskan alur pemesanan barang |
| *Pre Condition* | : | Form pengolah data pesanan terbuka |
| *Post Condition* | : | Data pesanan akan tersimpan di dalam table pesanan |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor mengisikan data pesanan yang dipesan |  | |
| 1. Klik tombol SIMPAN |  | |
|  | 1. Sistem akan melakukan pengecekan data pesanan yang diisikan apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka sistem akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI PEMESANAN ANDA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data pesanan akan tersimpan di tabel pesanan | |
| Alternatif : *update* data pesanan |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih data yang akan dirubah dan melakukan perubahan data |  | |
| 1. Klik tombol UPDATE |  | |
|  | 1. Sistem akan melakukan pengecekan data yang diubah apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka system akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA PEMESANAN ANDA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data pesanan akan tersimpan di tabel pesanan | |
| Alternatif : *delete* data pesanan |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih pesanan yang akan dihapus |  | |
| 1. Klik tombol HAPUS |  | |
|  | 1. Sistem akan menampilkan *message* “YAKIN DATA INI AKAN DIHAPUS ?” | |
| 1. Aktor klik button YA |  | |
|  | 1. Data yang dihapus akan hilang dari tabel pesanan | |

* Deskripsi *Use Case* Olah Data *User*

**Tabel 3.21 Deskripsi *Use Case* Olah Data *User***

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Pesanan** |
| Aktor | : | Member dan pengunjung |
| *Description* | : | *Use case* ini digunakan untuk menjelaskan alur dari olah data *user* |
| *Pre Condition* | : | Form olah data user ditampilkan |
| *Post Condition* | : | Data *user* akan tersimpan di dalam tabel *user* |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor akan mengisikan data *user* yang akan ditambah |  | |
| 1. Klik tombol Tambah |  | |
|  | 1. Sistem akan melakukan pengecekan data *user* yang diisikan apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka sistem akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data *user* akan tersimpan di tabel *user* | |
| Alternatif : *update* data *user* |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih data yang akan dirubah dan melakukan perubahan data |  | |
| 1. Klik tombol UPDATE |  | |
|  | 1. Sistem akan melakukan pengecekan data yang diubah apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka system akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data *user* akan tersimpan di tabel *user* | |
| Alternatif : *delete* data *user* |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih *user* yang akan dihapus |  | |
| 1. Klik tombol HAPUS |  | |
|  | 1. Sistem akan menampilkan *message* “YAKIN DATA INI AKAN DIHAPUS ?” | |
| 1. Aktor klik button YA |  | |
|  | 1. Data yang dihapus akan hilang dari tabel *user* | |

* Deskripsi *Use Case* Olah Data *Custom*

**Tabel 3.22 Deskripsi *Use Case* Olah Data *Custom***

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Pesanan** |
| Aktor | : | Member dan pengunjung |
| *Description* | : | *Use case* ini digunakan untuk menjelaskan alur dari olah data *custom* |
| *Pre Condition* | : | Form olah data *custom* ditampilkan |
| *Post Condition* | : | Data *custom* akan tersimpan di dalam tabel *custom* |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor akan mengisikan data *custom* yang akan ditambah |  | |
| 1. Klik tombol Tambah |  | |
|  | 1. Sistem akan melakukan pengecekan data *custom* yang diisikan apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka sistem akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data *custom* akan tersimpan di tabel *custom* | |
| Alternatif : *update* data *custom* |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih data yang akan dirubah dan melakukan perubahan data |  | |
| 1. Klik tombol UPDATE |  | |
|  | 1. Sistem akan melakukan pengecekan data yang diubah apakah telah sesuai atau belum | |
|  | 1. Jika ada yang belum diisi, maka system akan menampilkan *MESSAGE* “MAAF SILAHKAN LENGKAPI DATA” | |
|  | 1. Jika data yang diisikan telah sesuai, maka data *custom* akan tersimpan di tabel *custom* | |
| Alternatif : *delete* data *custom* |  | |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor memilih *custom* yang akan dihapus |  | |
| 1. Klik tombol HAPUS |  | |
|  | 1. Sistem akan menampilkan *message* “YAKIN DATA INI AKAN DIHAPUS ?” | |
| 1. Aktor klik button YA |  | |
|  | 1. Data yang dihapus akan hilang dari tabel *custom* | |

* Deskripsi *Use Case* Laporan Persediaan Produk

**Tabel 3.23 Deskripsi *Use Case* Laporan Persediaan Produk**

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Pesanan** |
| Aktor | : | Admin |
| *Description* | : | *Use case* ini digunakan untuk menjelaskan alur dari pelaporan persediaan produk |
| *Pre Condition* | : | Form laporan persediaan produk ditampilkan |
| *Post Condition* | : | Mencetak laporan persediaan produk |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor melakukan persiapan pencetakan laporan persediaan produk di dalam halaman laporan |  | |
| 1. Klik tombol CETAK |  | |
|  | 1. Sistem akan menampilkan laporan persediaan produk yang akan dicetak | |
|  | 1. Mencetak laporan persediaan produk | |

* Deskripsi *Use Case* Laporan Penjualan

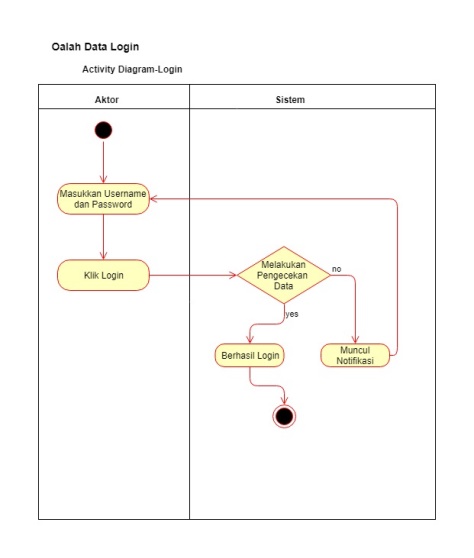
**Tabel 3.23 Deskripsi *Use Case* Laporan Penjualan**

|  |  |  |
| --- | --- | --- |
| **Nama Use Case** | **:** | **Olah Data Pesanan** |
| Aktor | : | Admin |
| *Description* | : | *Use case* ini digunakan untuk menjelaskan alur dari pelaporan penjualan |
| *Pre Condition* | : | Form laporan penjualan ditampilkan |
| *Post Condition* | : | Mencetak laporan penjualan |
| **Aksi Sistem** | **Reaksi Sistem** | |
| 1. Aktor melakukan persiapan pencetakan laporan penjualan di dalam halaman laporan |  | |
| 1. Klik tombol CETAK |  | |
|  | 1. Sistem akan menampilkan laporan penjualan yang akan dicetak | |
|  | 1. Mencetak laporan penjualan | |

**3.6.1 *Activity* Diagram**

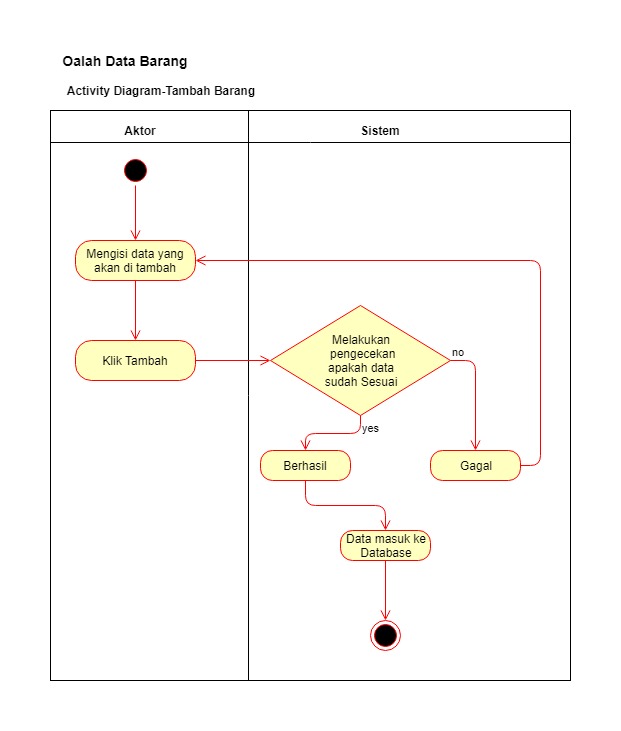
*Activity* diagram atau diagram aktivitasadalah bentuk visual dari alur kerja yang berisi aktivitas dan tindakan, yang juga dapat berisi pilihan, pengulangan, dan *concurrency*. Dalam UML, diagram aktivitas dibuat untuk menjelaskan aktivitas komputer maupun alur aktivitas dalam organisasi. Diagram aktivitas menggambarkan alur kontrol secara garis besar.

* *Activity* DiagramOlah Data Login

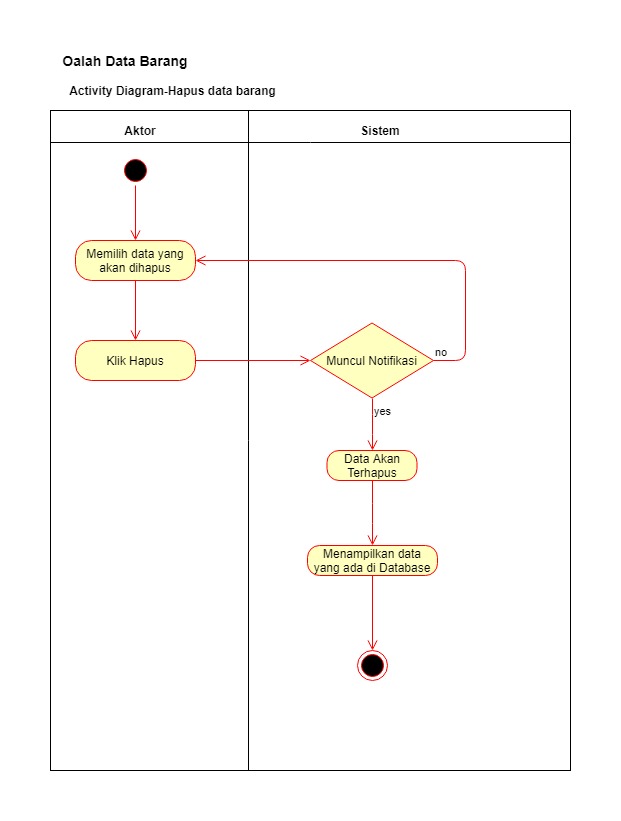
**

**Gambar 3.22 *Activiy* DiagramProses Input Data Kategori**

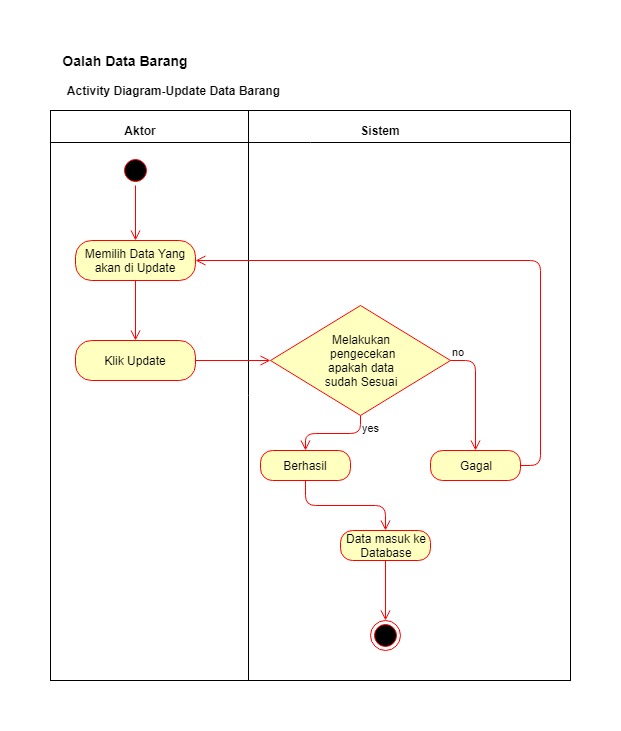
* *Activity* DiagramOlah Data Login

**

**Gambar 3.23 *Activiy* DiagramProses Input Data Barang**

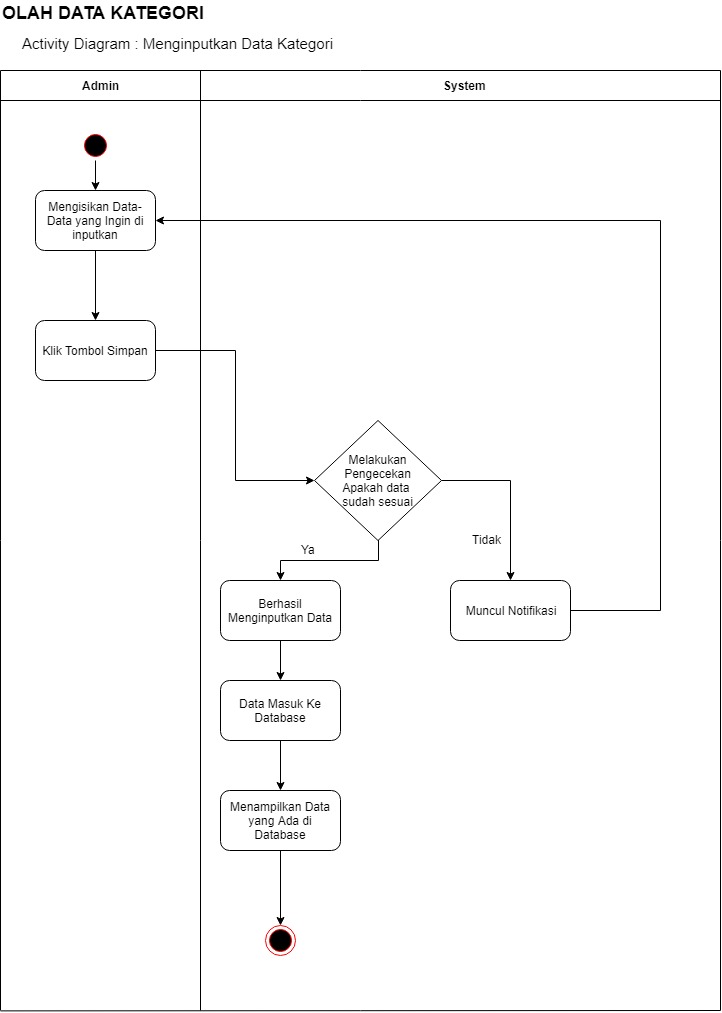
**

**Gambar 3.24 *Activiy* DiagramProses *Delete* Data Barang**

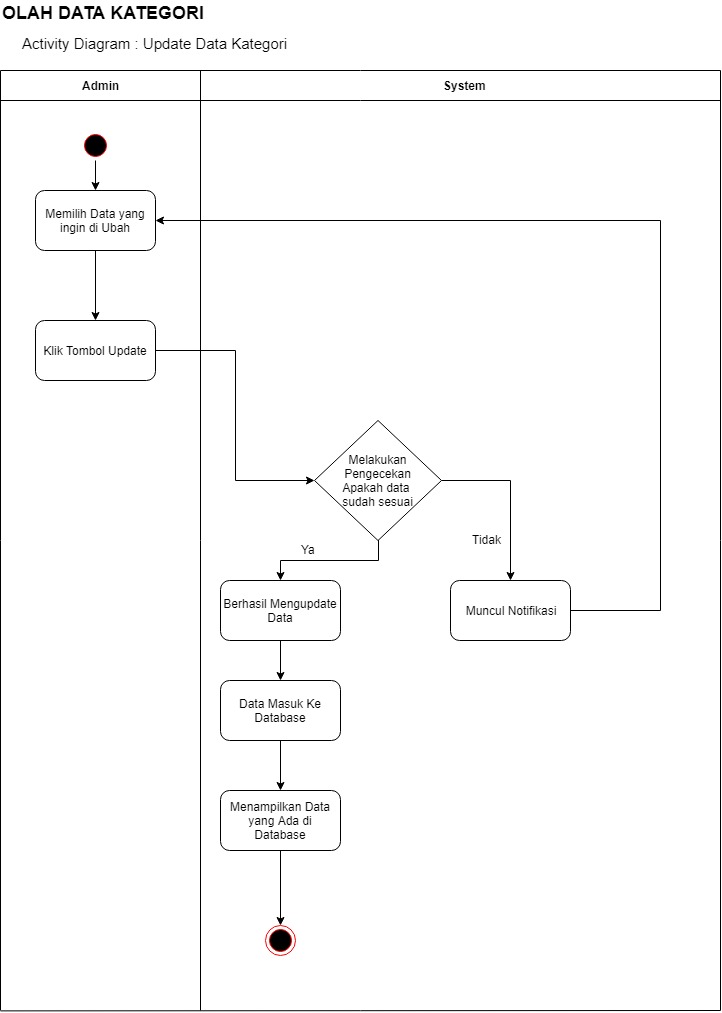
**

**Gambar 3.25 *Activiy* DiagramProses *Update* Data Barang**

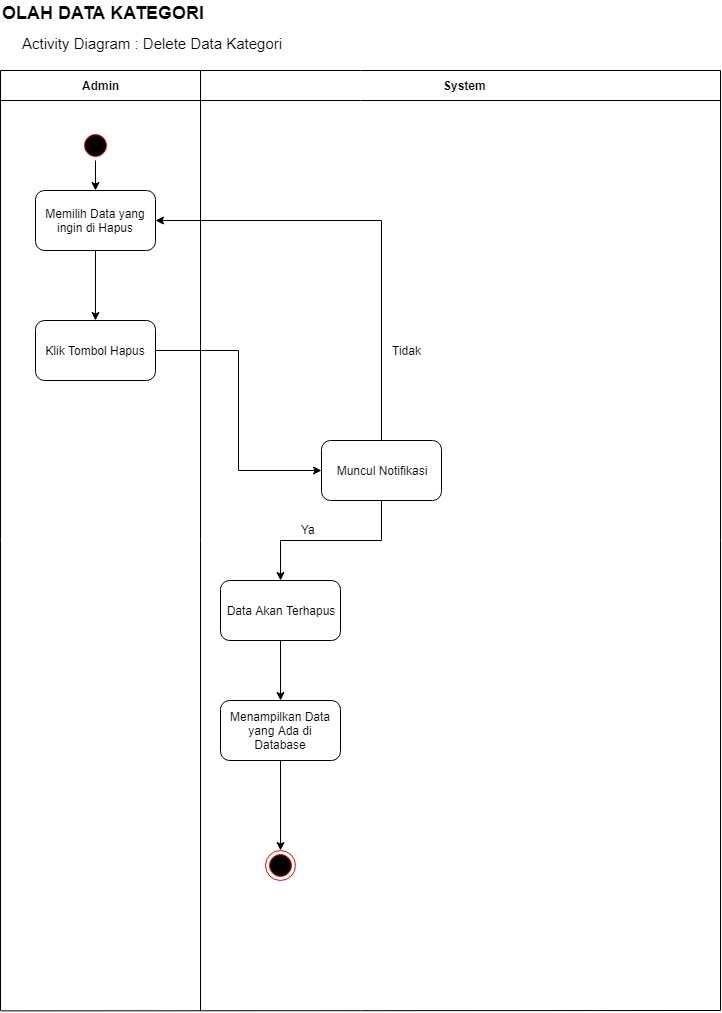
* *Activity* DiagramOlah Data Kategori



**Gambar 3.26 *Activiy* DiagramProses Input Data Kategori**

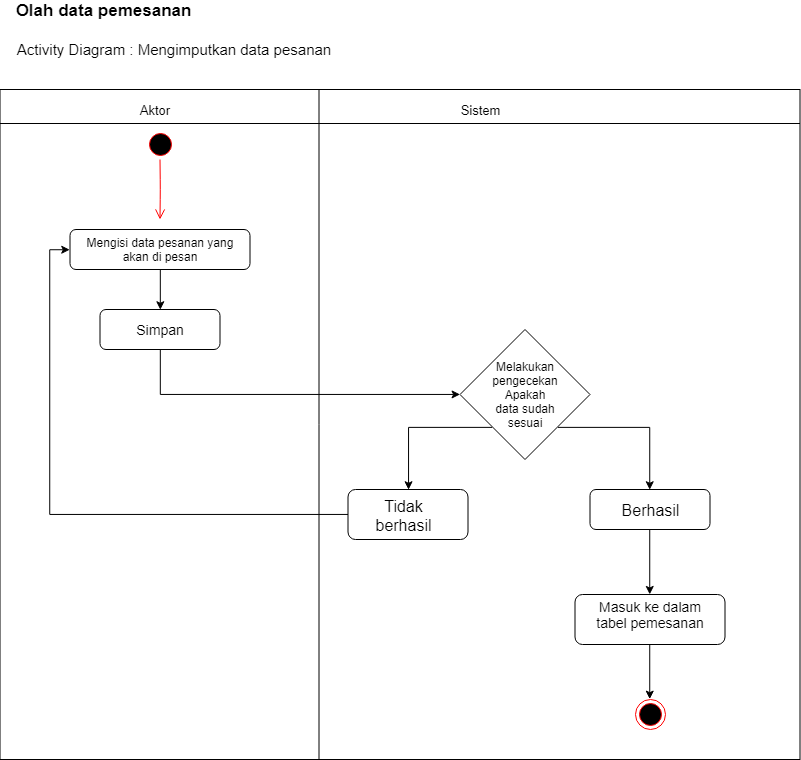


**Gambar 3.27 *Activiy* DiagramProses *Update* Data Kategori**

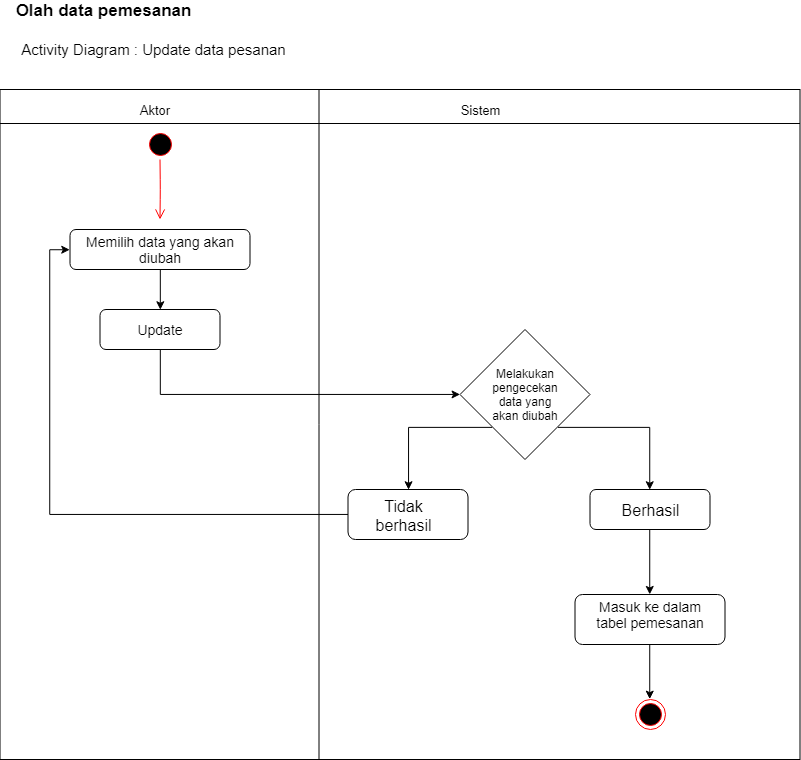


**Gambar 3.28 *Activiy* DiagramProses *Delete* Data Kategori**

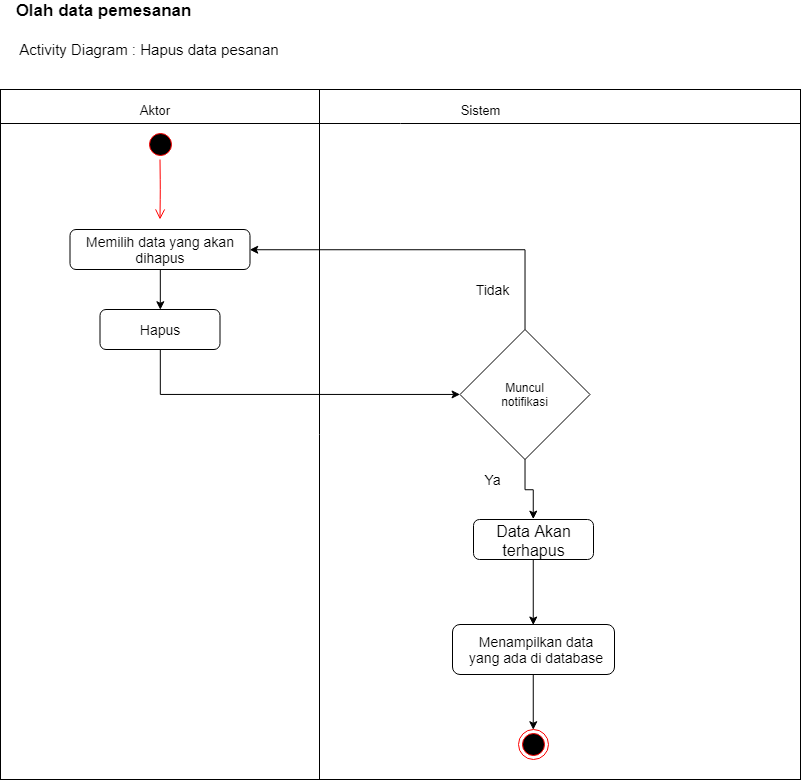
* *Activity Diagram* Olah Data Pesanan



**Gambar 3.29 *Activiy* DiagramProses Input Data Pesanan**

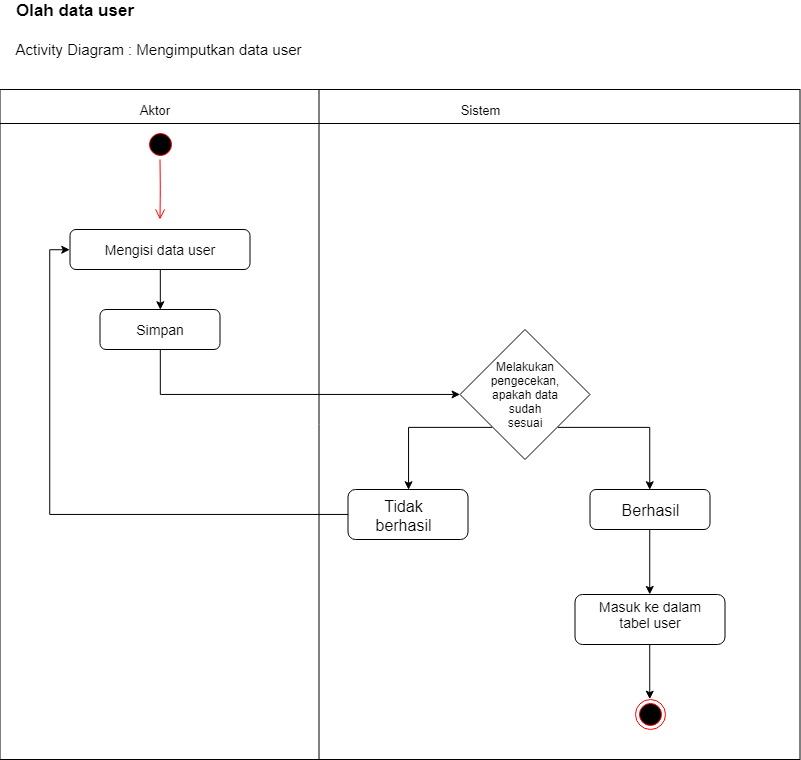


**Gambar 3.30 *Activiy* DiagramProses *Update* Data Pesanan**

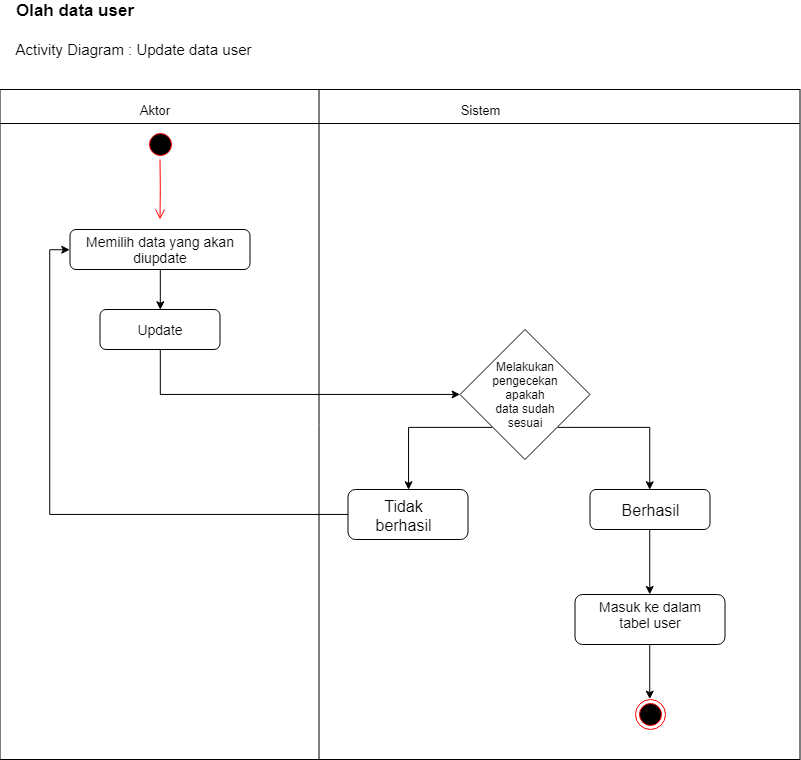


**Gambar 3.31 *Activiy* DiagramProses *Delete* Data Pesanan**

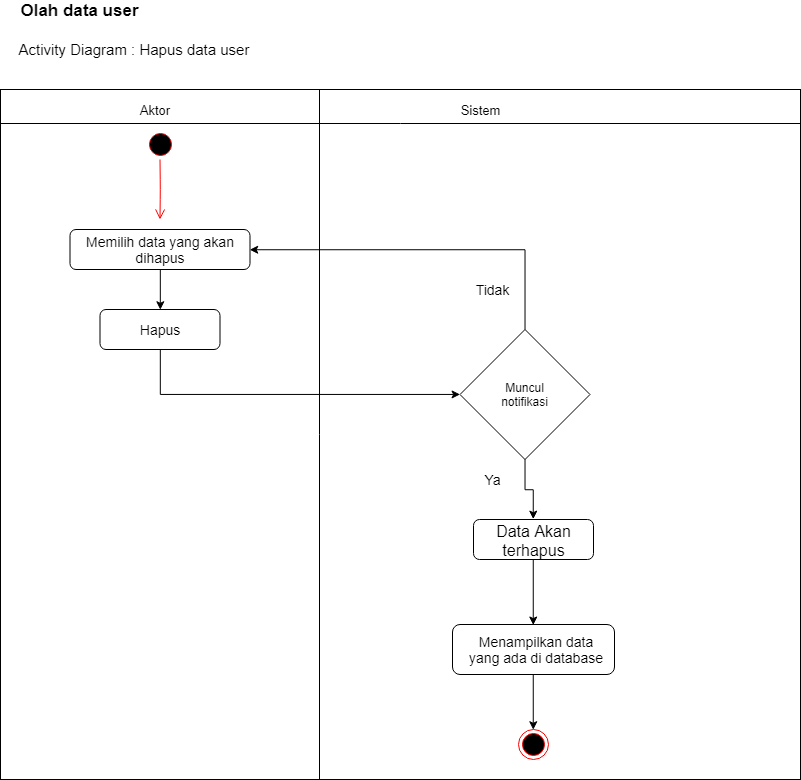
* *Activity* DiagramOlah Data *User*

**

**Gambar 3.32 *Activiy* DiagramProses Input Data *User***

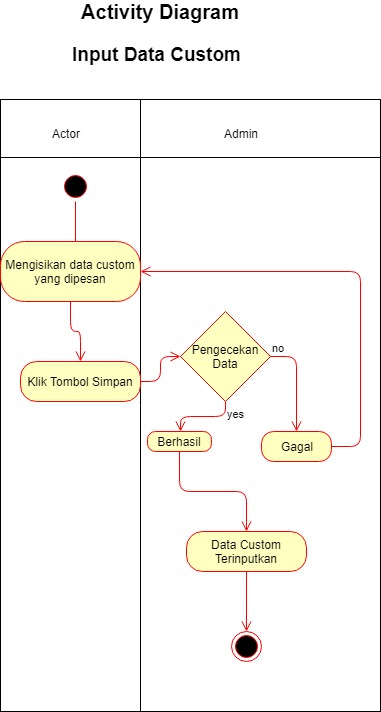


**Gambar 3.33 *Activiy* DiagramProses *Update* Data *User***

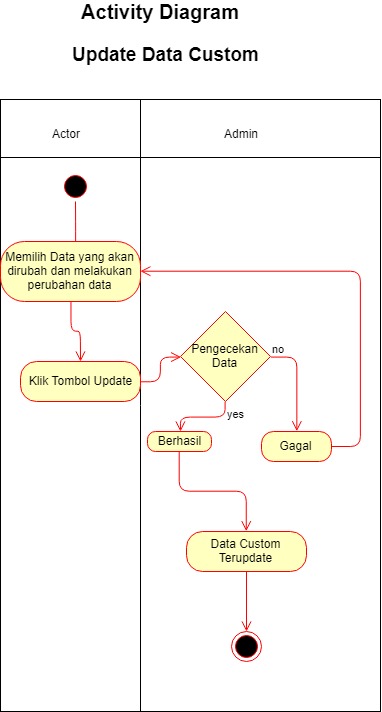


**Gambar 3.34 *Activiy* DiagramProses *Delete* Data *User***

* *Activity* DiagramOlah Data *Custom*

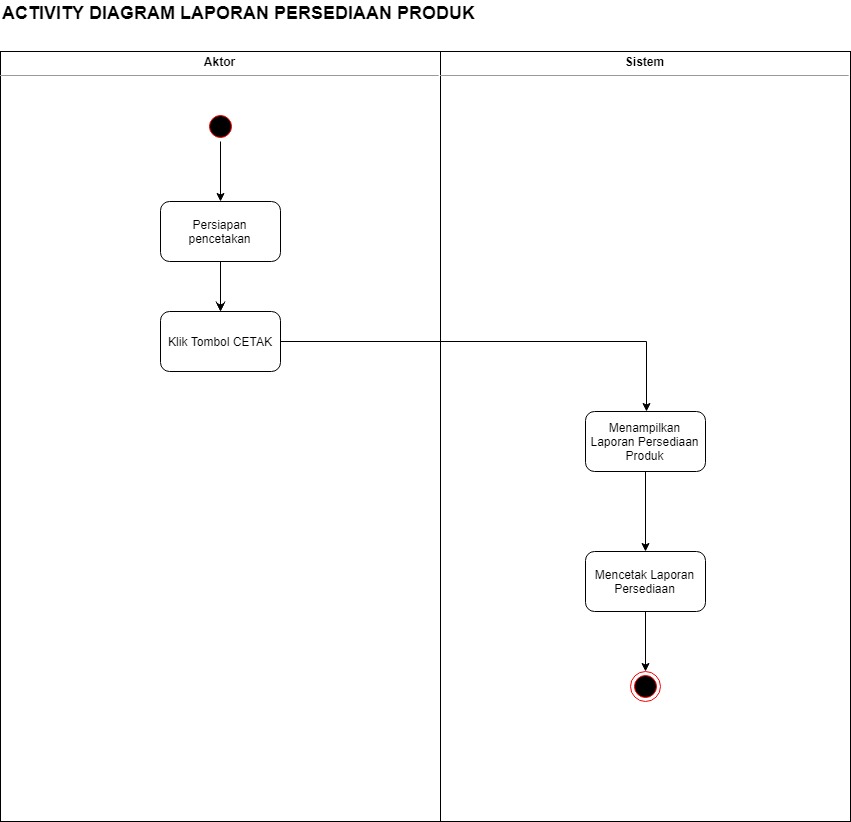
**

**Gambar 3.35 *Activiy* DiagramProses Input Data *Custom***



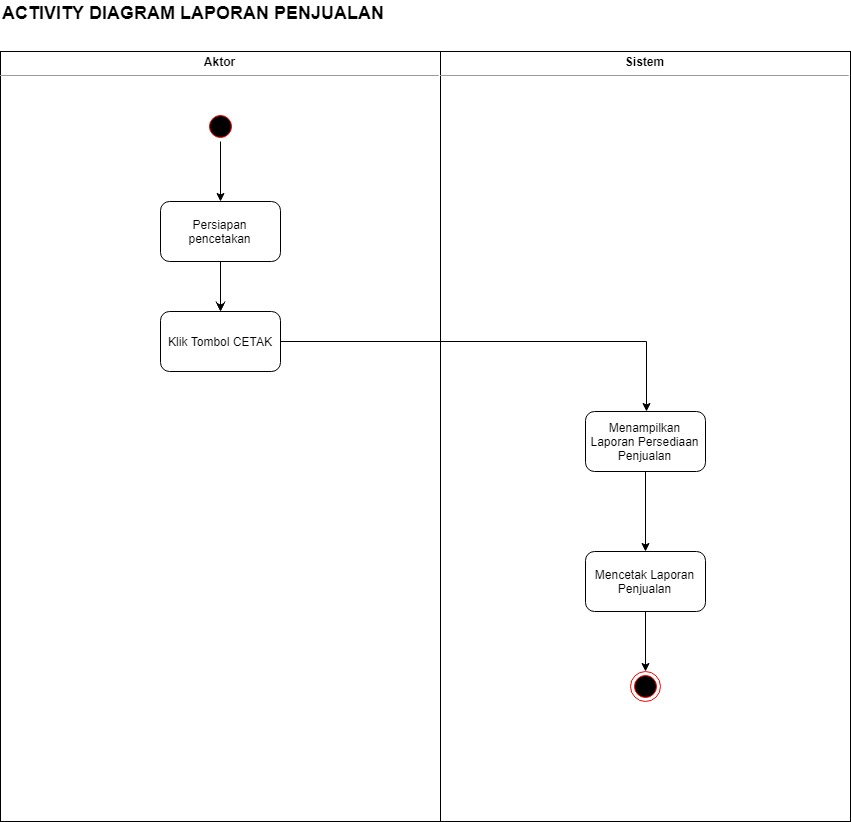
**Gambar 3.36 *Activiy* DiagramProses *Update* Data *Custom***

* *Activity* DiagramLaporan Persediaan Produk

**

**Gambar 3.37 *Activity* DiagramLaporan Persediaan Produk**

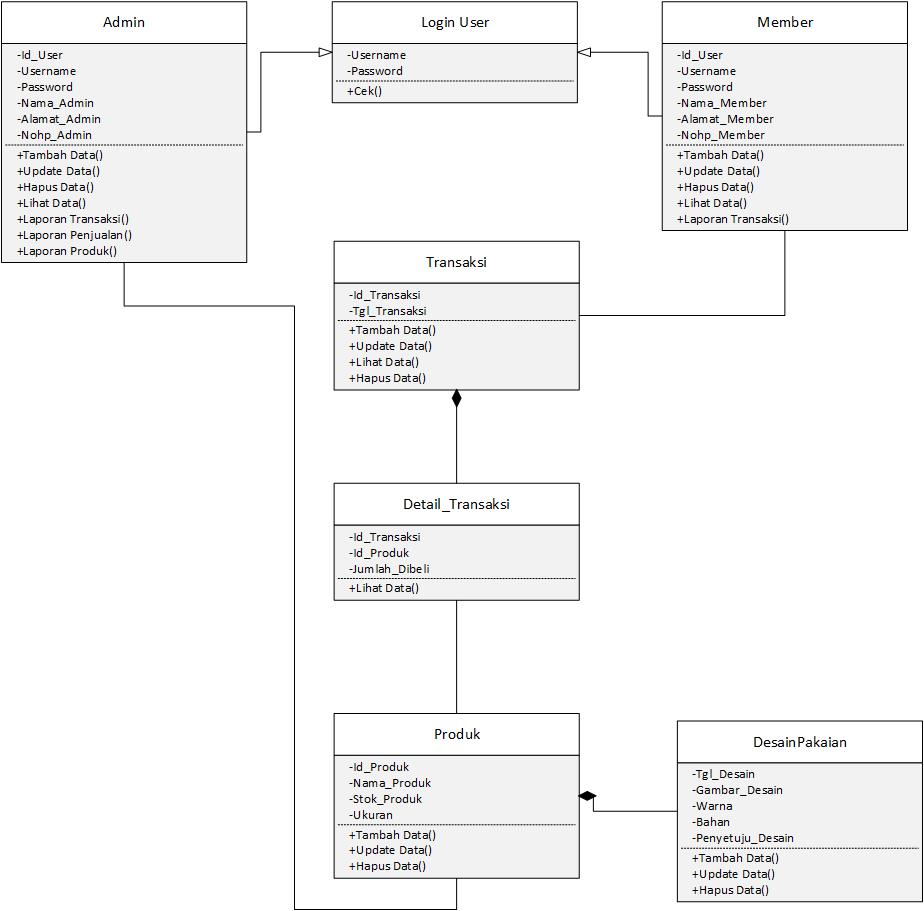
* *Activity* DiagramLaporan Penjualan



**Gambar 3.38 *Activity* DiagramLaporan Penjualan**

**3.6.3 *Class* Diagram**

*Class* diagramatau kelas diagram adalah diagram UML yang menggambarkan kelas-kelas dalam sebuah sistem dan hubungannya antara satu dengan yang lain, serta dimasukkan pula atribut dan operasi. Berikut ini adalah gambar kelas diagram sistem informasi penjualan Toko Sablon Kaos Ngapak.

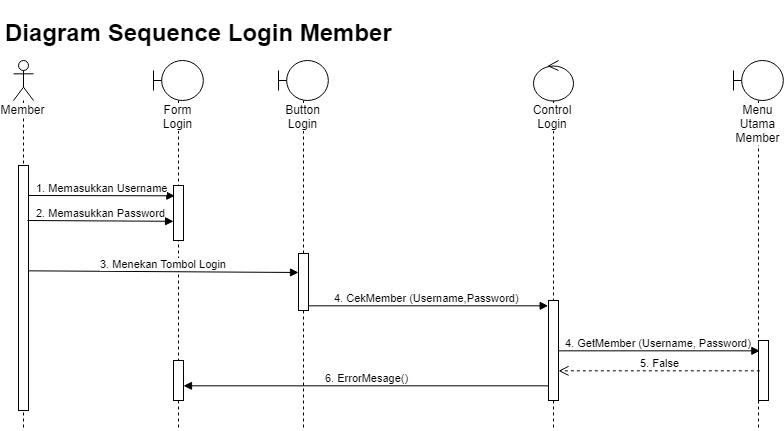


**Gambar 3.39 *Activity* Diagram Laporan Penjualan**

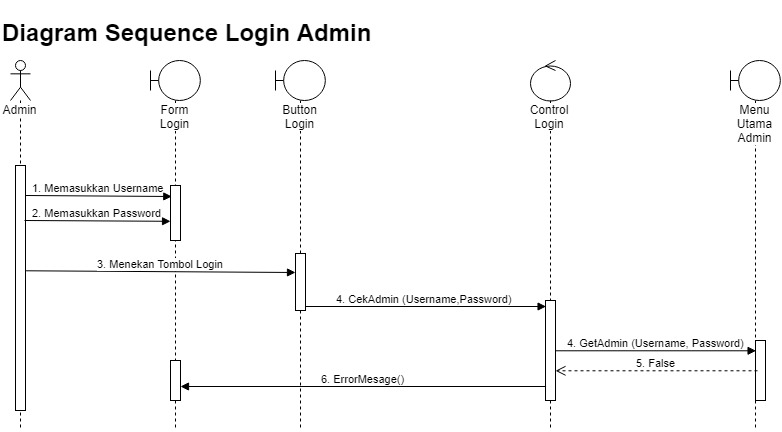
**3.6.4 *Sequence* Diagram**

*Sequence* diagram adalah salah satu dari diagram-diagram yang ada pada UML, *sequence* diagram ini adalah diagram yang menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*. Sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem. Berikut ini adalah *sequence* diagramyang terdapat pada sistem.

* *Login*

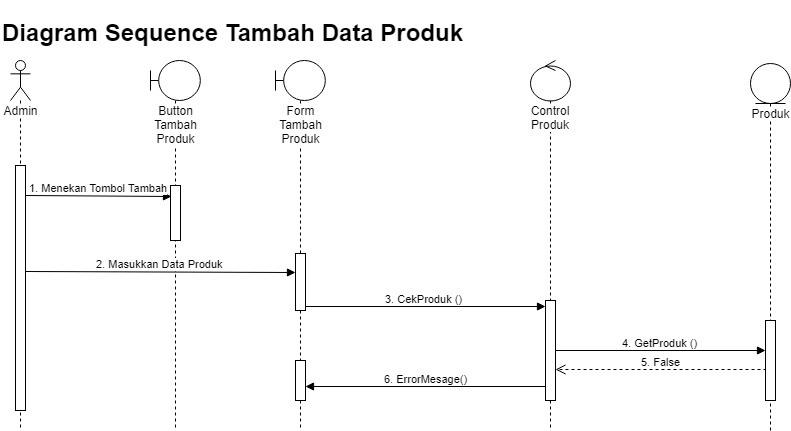


**Gambar 3.40 *Sequence* Diagram *Login* Member**

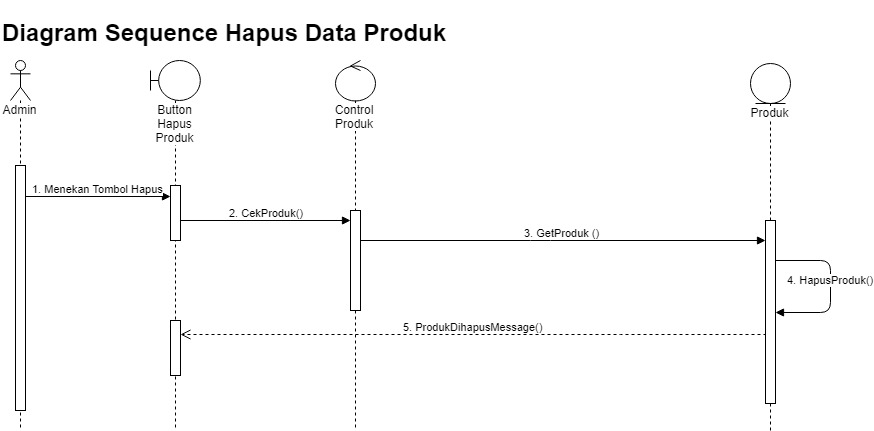


**Gambar 3.41 *Sequence* Diagram *Login* Admin**

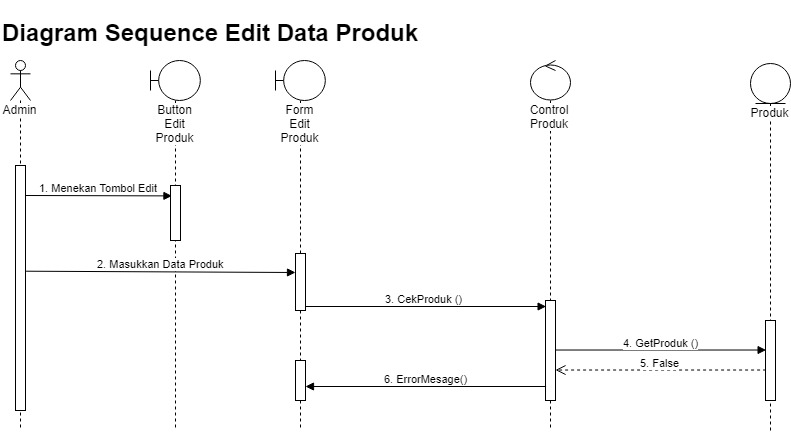
* Produk



**Gambar 3.42 *Sequence* DiagramTambah Data Produk**

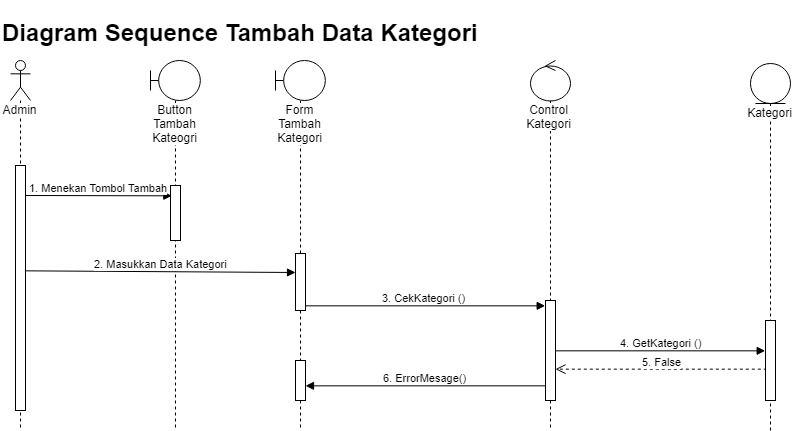


**Gambar 3.43 *Sequence* DiagramHapus Data Produk**

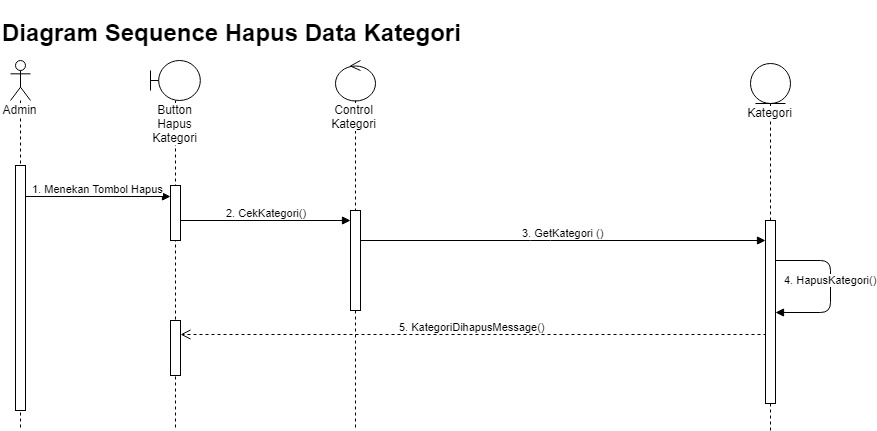


**Gambar 3.44 *Sequence* DiagramEdit Data Produk**

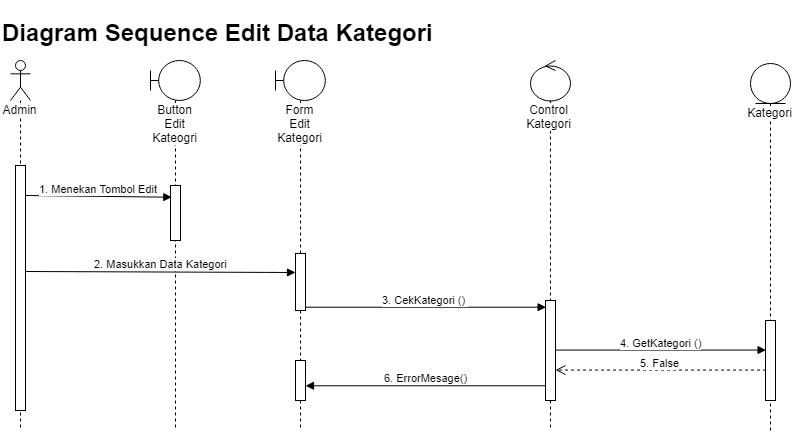
* Kategori

**

**Gambar 3.45 *Sequence* DiagramTambah Data Kategori**

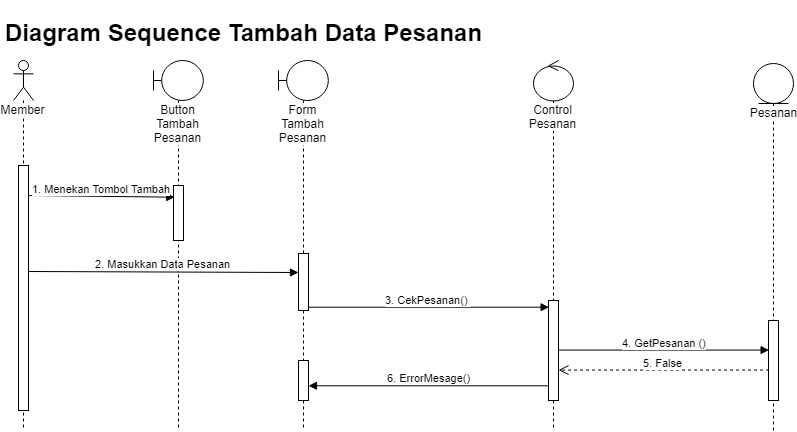
**

**Gambar 3.46 *Sequence* DiagramHapus Data Kategori**

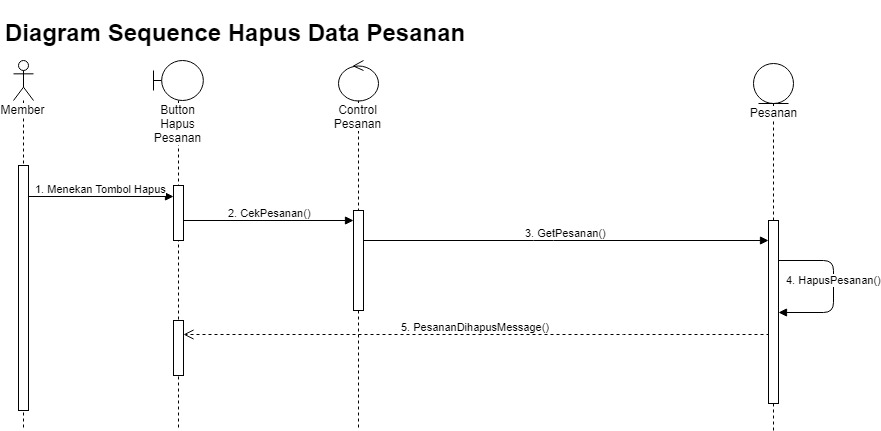
**

**Gambar 3.47 *Sequence* DiagramEdit Data Kategori**

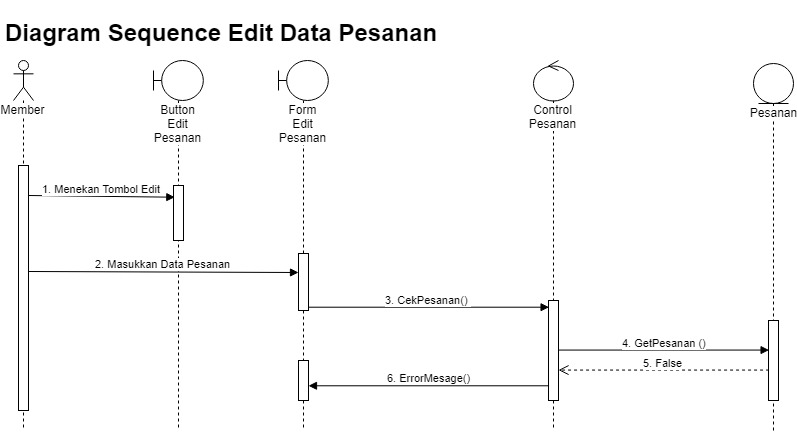
* Pesanan

**

**Gambar 3.48 *Sequence* DiagramTambah Data Pesanan**

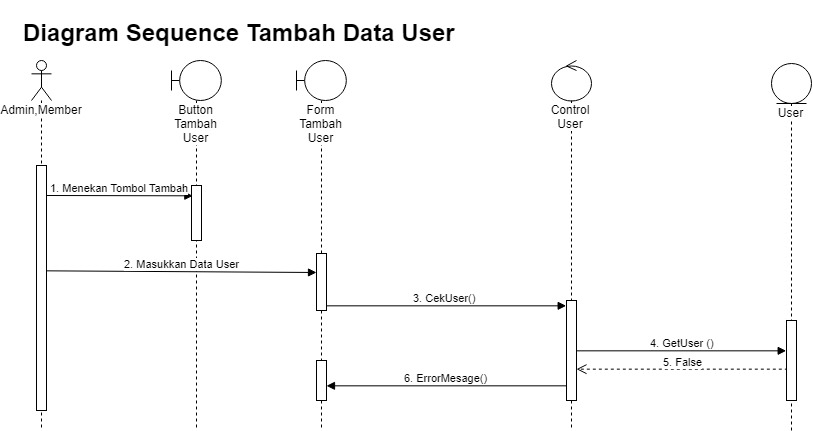
**

**Gambar 3.49 *Sequence* DiagramHapus Data Pesanan**

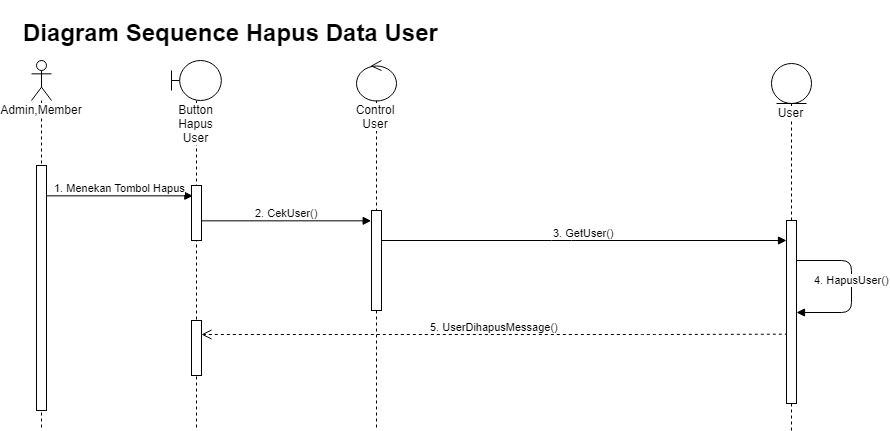
**

**Gambar 3.50 *Sequence* DiagramEdit Data Pesanan**

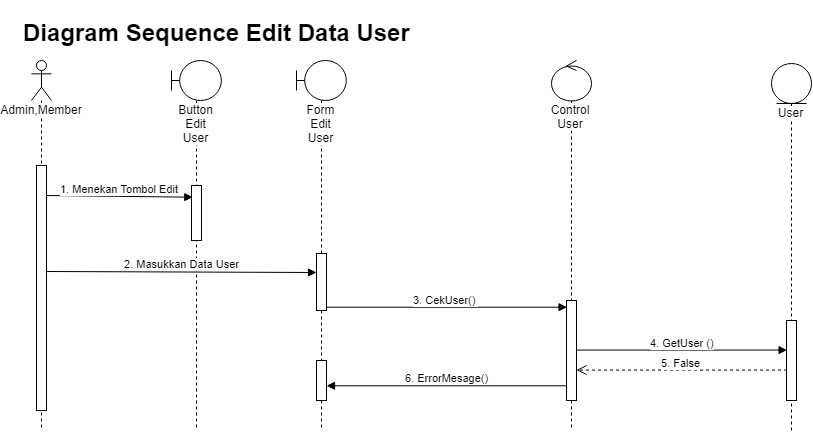
* *User*

**

**Gambar 3.51 *Sequence* DiagramTambah Data *User***

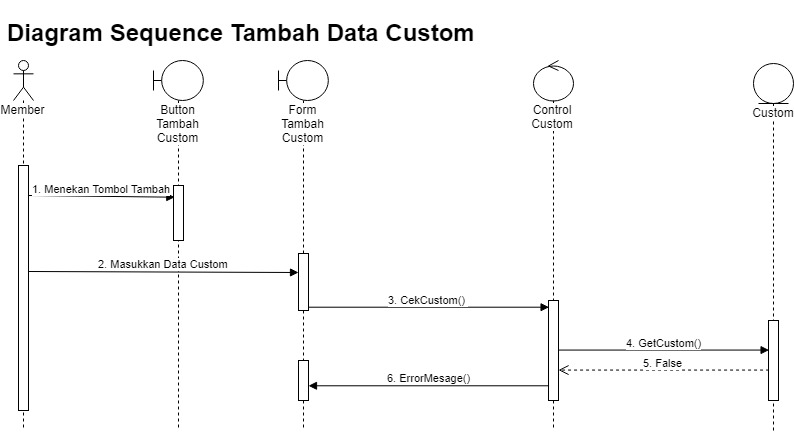
**

**Gambar 3.52 *Sequence* DiagramHapus Data *User***

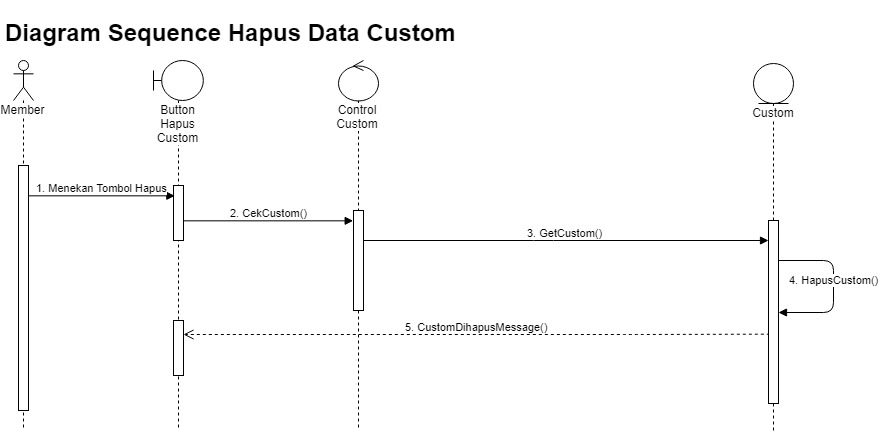
**

**Gambar 3.53 *Sequence* DiagramEdit Data *User***

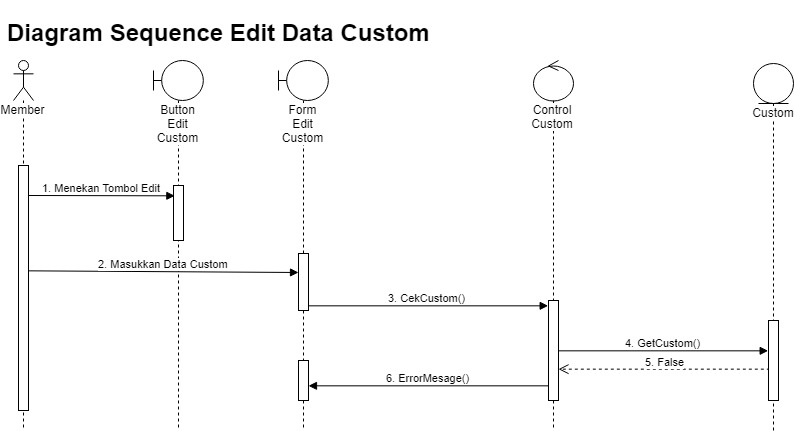
* *Custom*

**

**Gambar 3.54 *Sequence* DiagramTambah Data *Custom***

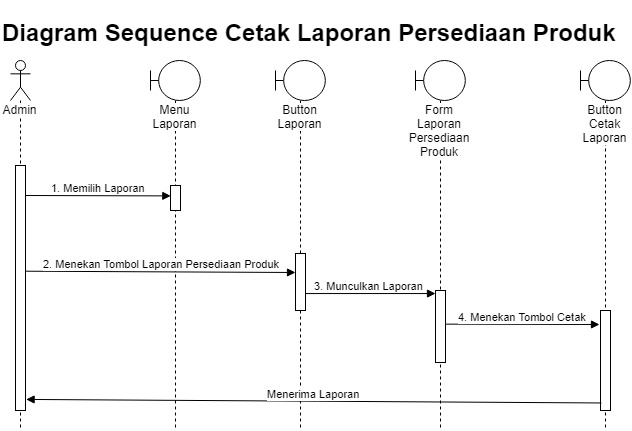
**

**Gambar 3.55 *Sequence* DiagramHapus Data *Custom***

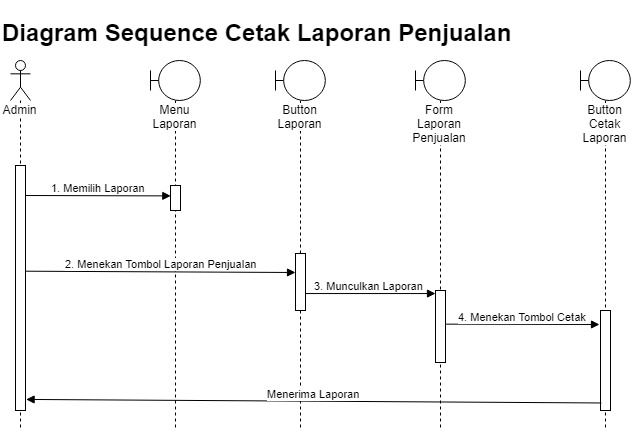
**

**Gambar 3.56 *Sequence* DiagramEdit Data *Custom***

* Laporan



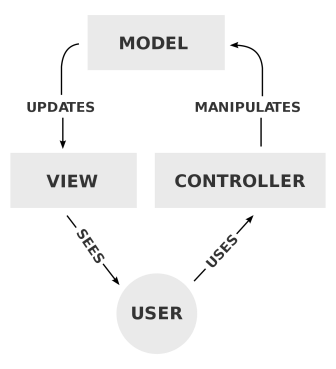
**Gambar 3.57 *Sequence* DiagramCetak Laporan Persediaan Produk**



**Gambar 3.58 *Sequence* DiagramCetak Laporan Penjualan**

**3.7 *Software* Arsitektur (MVC/MVP/MVvM)**

Dalam pembuatan sistem informasi penjualan online Toko Sablon Kaos Ngapak *software* arsitektur yang digunakan adalah *Model-View-Controller* atau MVC. MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*). Dalam implementasinya kebanyakan *framework* dalam aplikasi *website* adalah berbasis arsitektur MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi *website*.



**Gambar 3.59 *Model-View Controller***

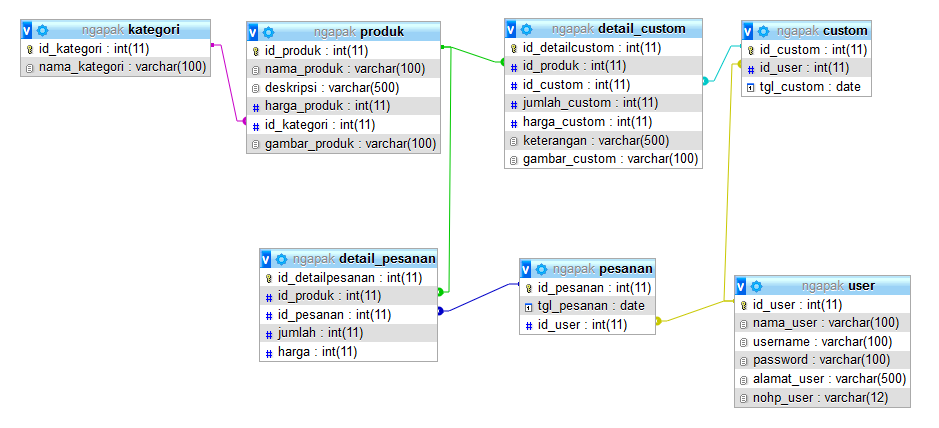
**BAB VI**

**IMPLEMENTASI DAN PEMBAHASAN**

**4.1 Implementasi**

**4.1.1 Implementasi Basis Data**

*Software* yang digunakan dalam melakukan implementasi racangan basis data adalah Apache Phmyadmin. Berikut ini adalah gambar relasi database yang telah diimplementasikan.

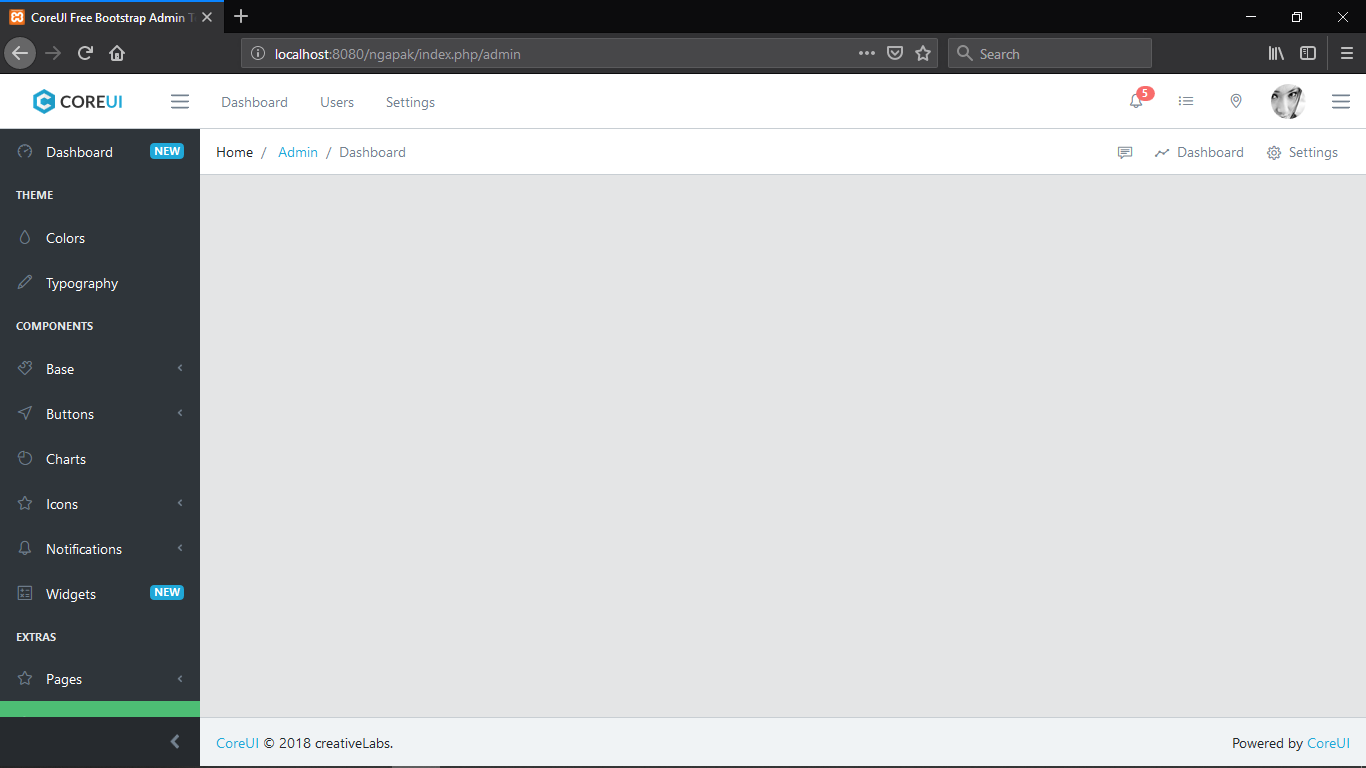


**Gambar 3.60 Implementasi Database**

**4.1.2 Implementasi Antar Muka**

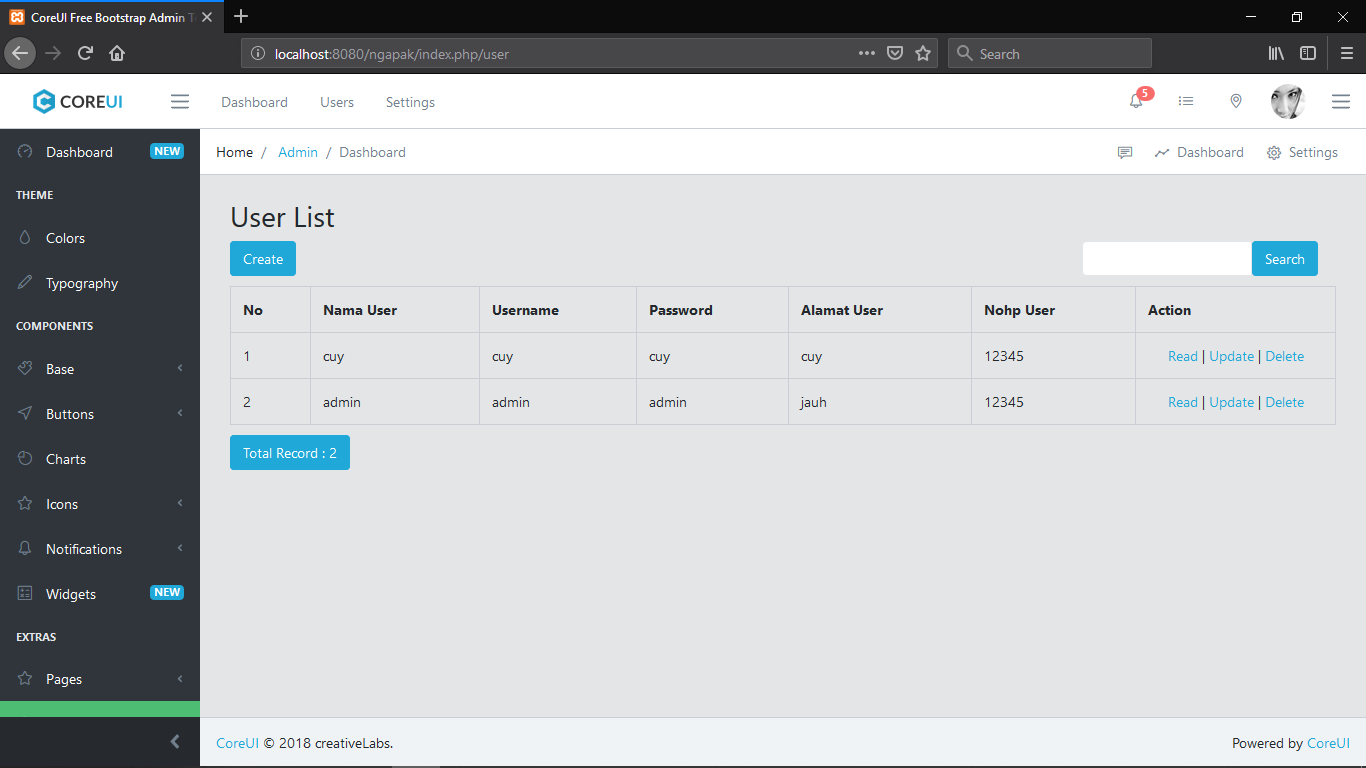
Implementasi Antar Muka yang telah berhasil adalah pada bagian pengelolaan data user, kategori dan produk. Berikut ini adalah gambarnya.

* Halaman Index Admin



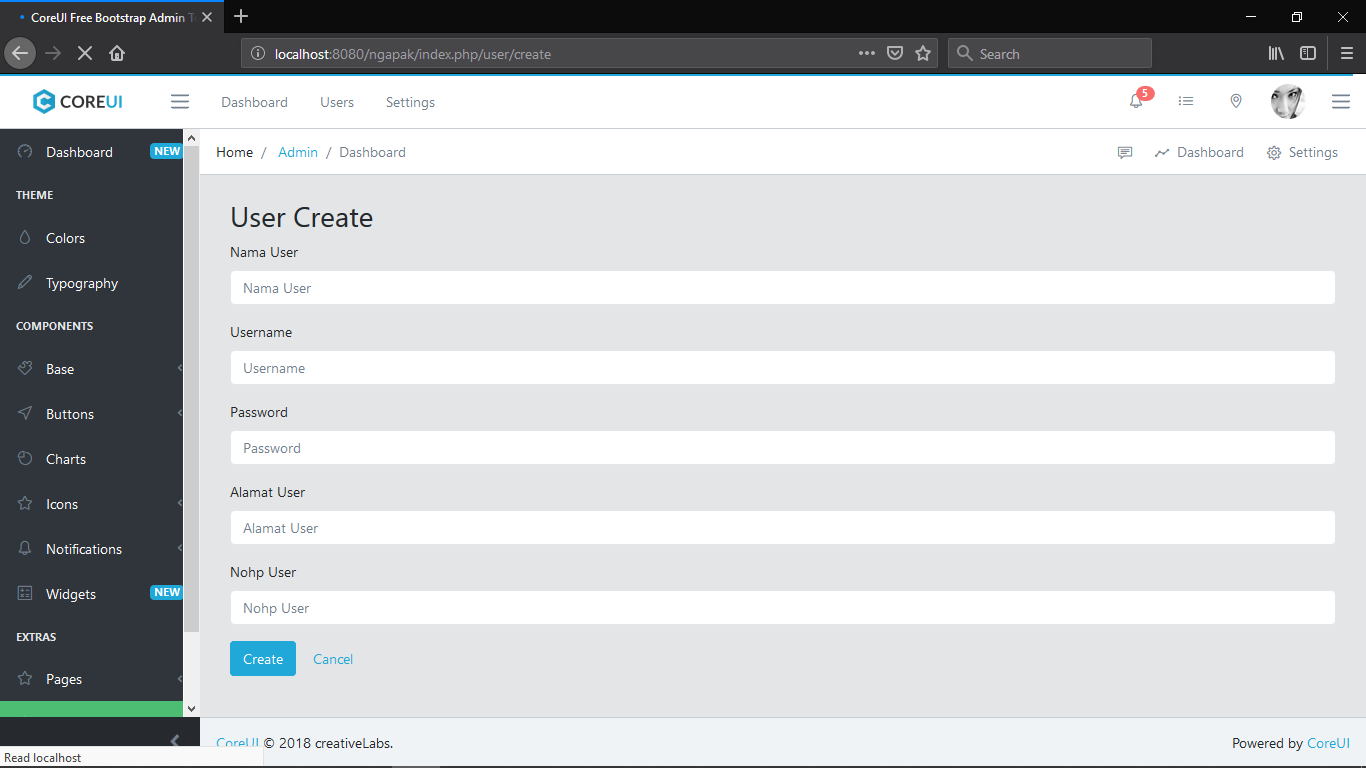
**Gambar 3.61 Halaman Index Admin**

* Halaman *List User* Admin



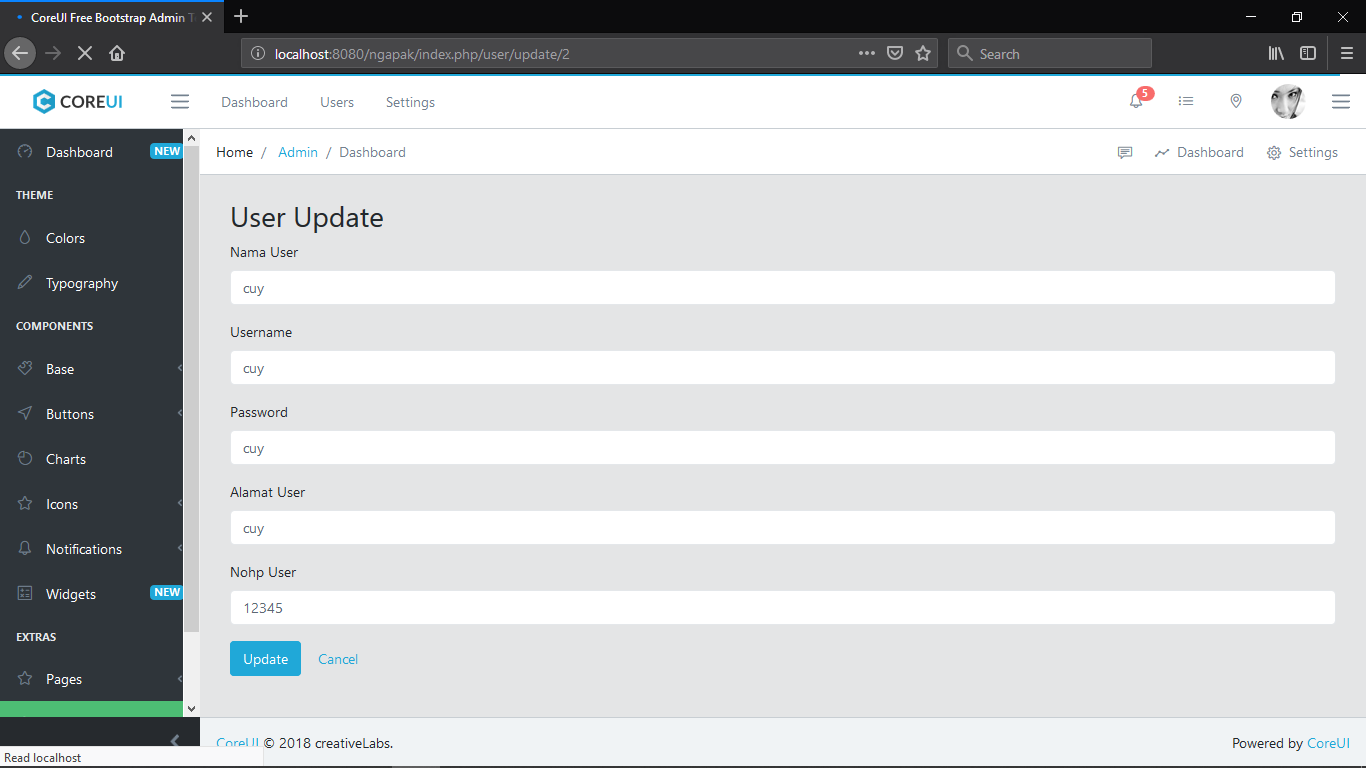
**Gambar 3.62 Halaman *List User* Admin**

* Halaman *Create User* Admin



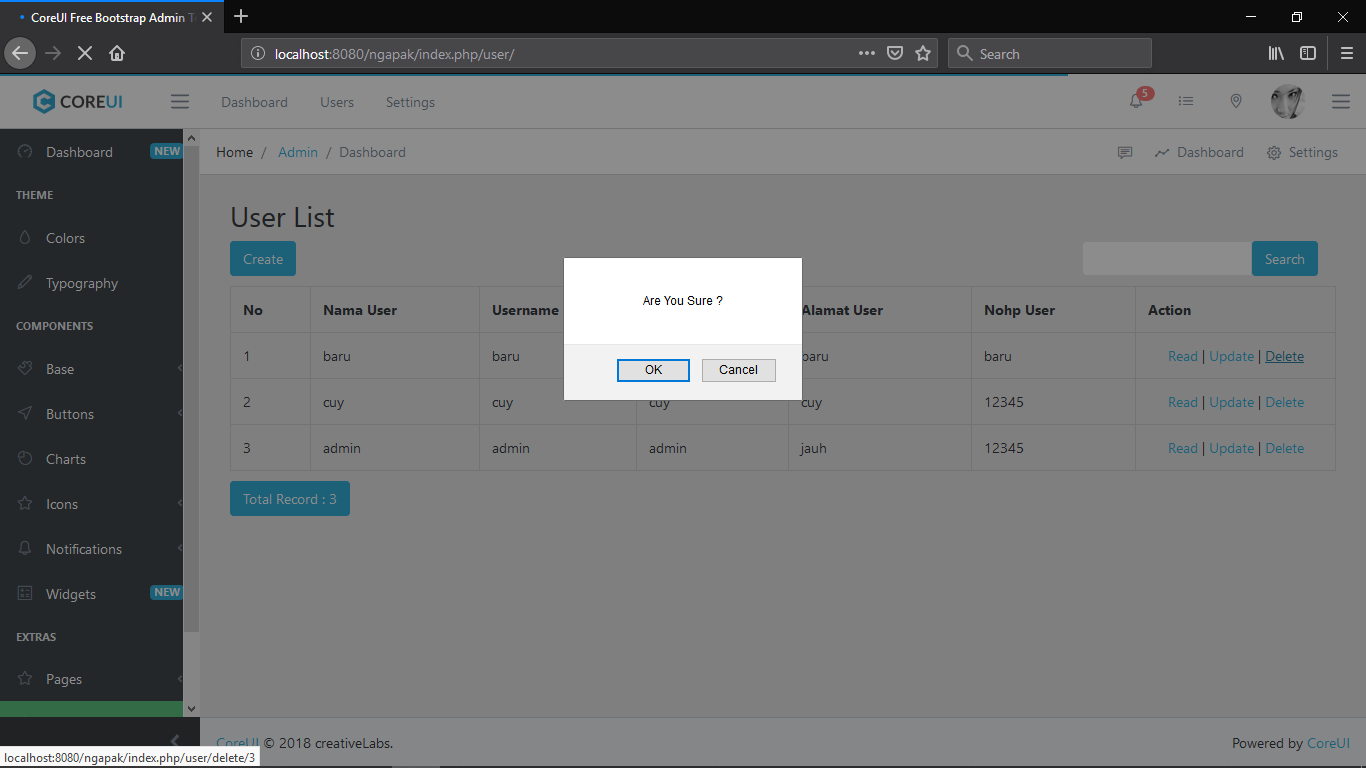
**Gambar 3.63 Halaman *Create User* Admin**

* Halaman *Update User* Admin



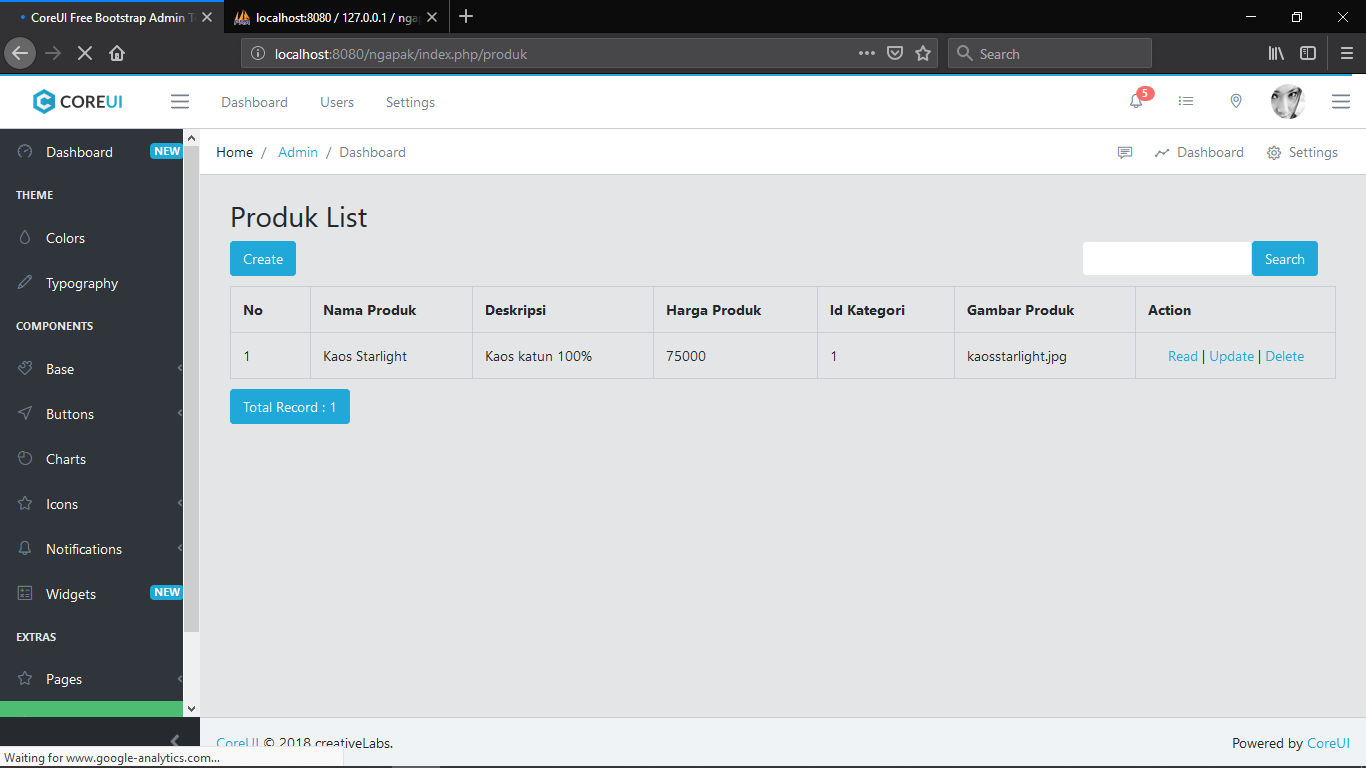
**Gambar 3.64 Halaman *Update User* Admin**

* Halaman *Delete User* Admin



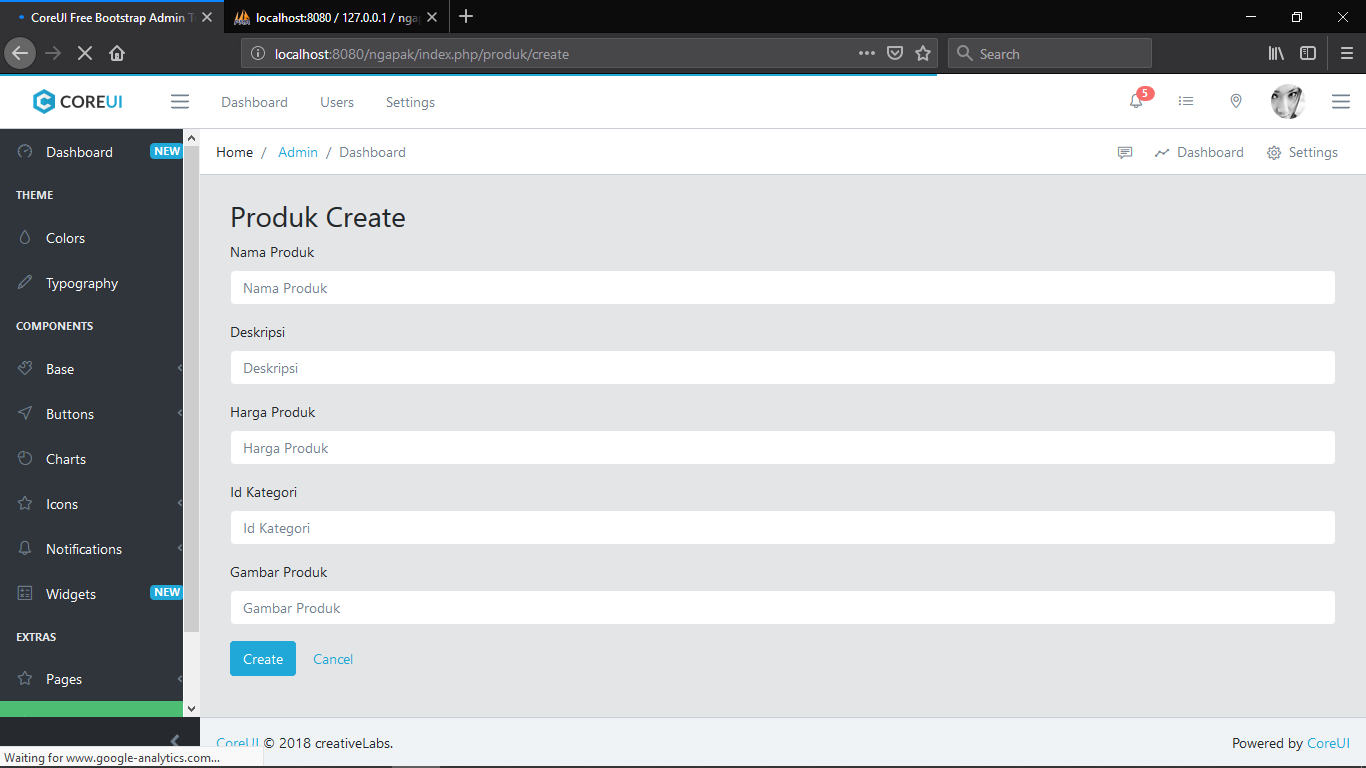
**Gambar 3.65 Halaman *Delete User* Admin**

* Halaman *List* ProdukAdmin



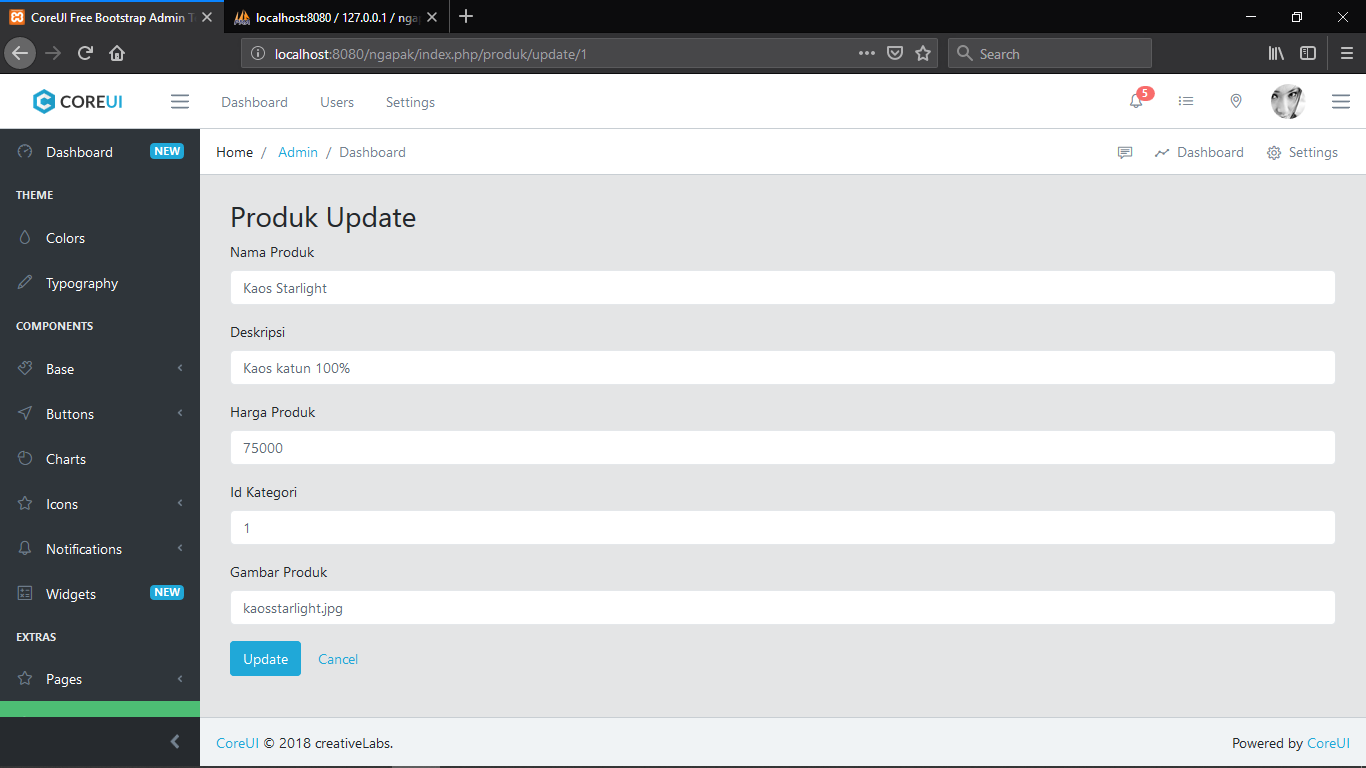
**Gambar 3.66 Halaman *List* Produk Admin**

* Halaman *Create* ProdukAdmin



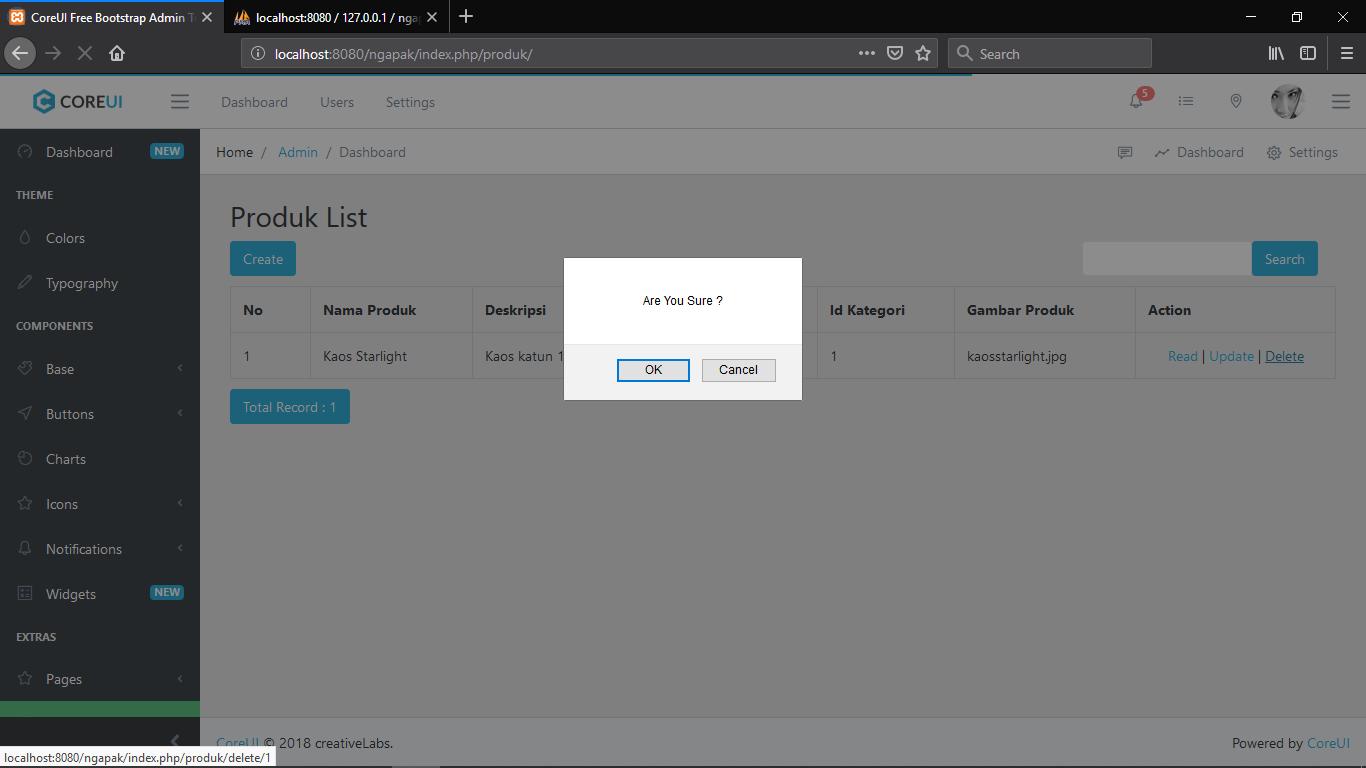
**Gambar 3.67 Halaman *Create* Produk Admin**

* Halaman *Update* ProdukAdmin



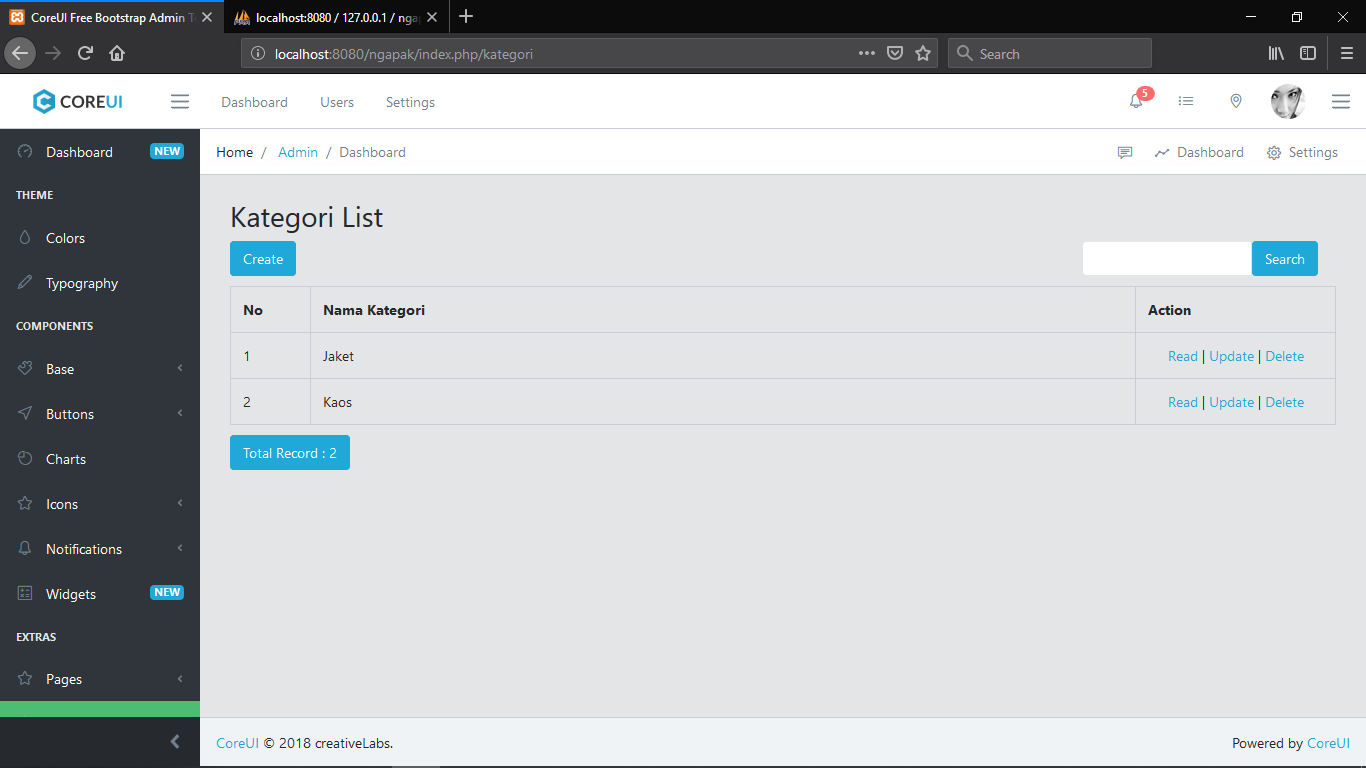
**Gambar 3.68 Halaman *Update* Produk Admin**

* Halaman *Delete* ProdukAdmin



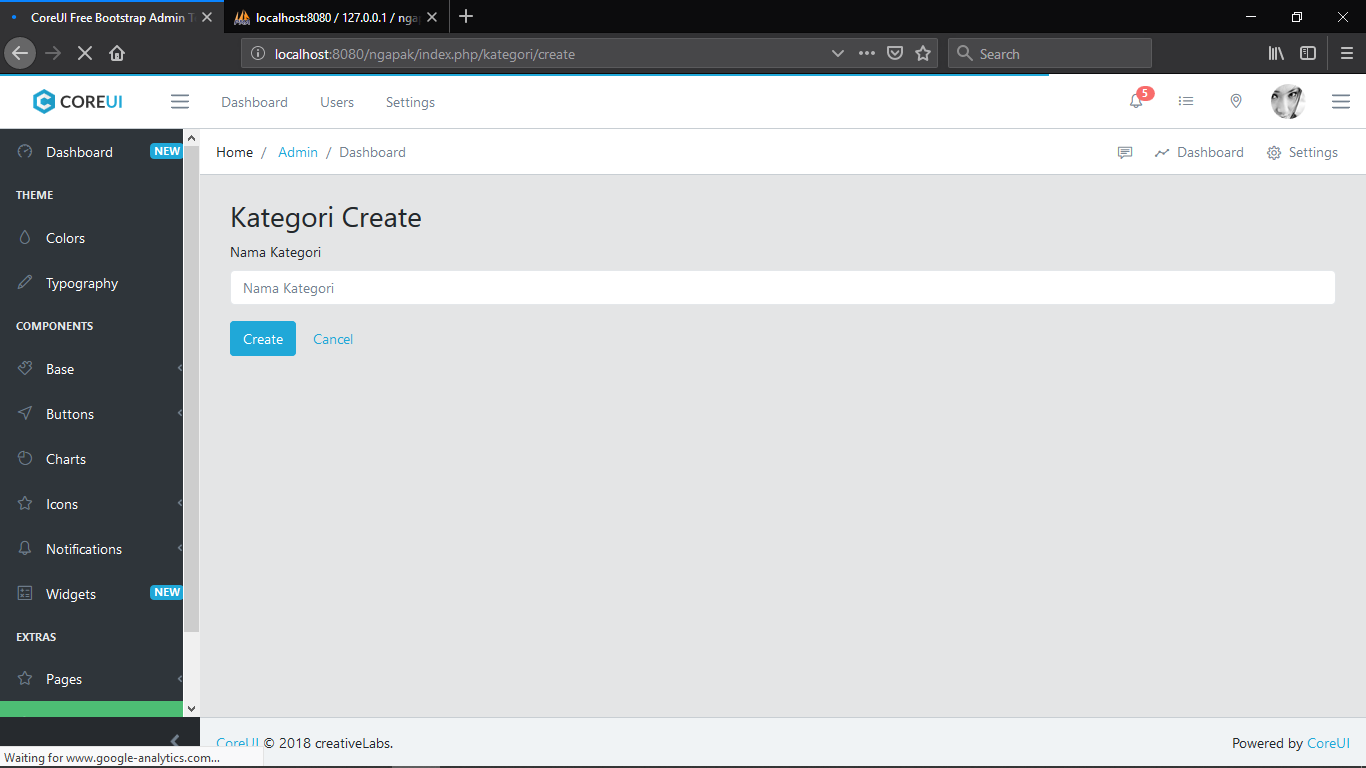
**Gambar 3.69 Halaman *Delete* Produk Admin**

* Halaman *List* KategoriAdmin



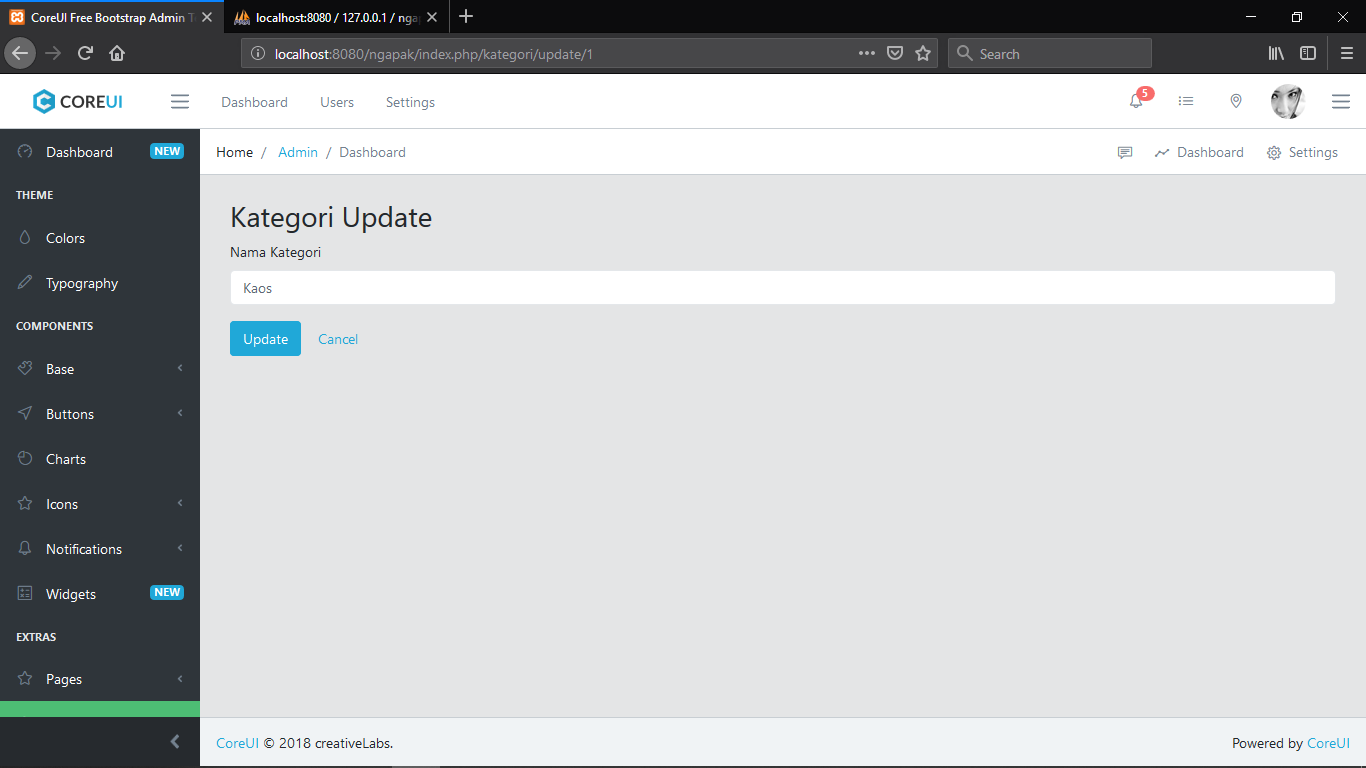
**Gambar 3.70 Halaman *List* Kategori Admin**

* Halaman *Create* KategoriAdmin



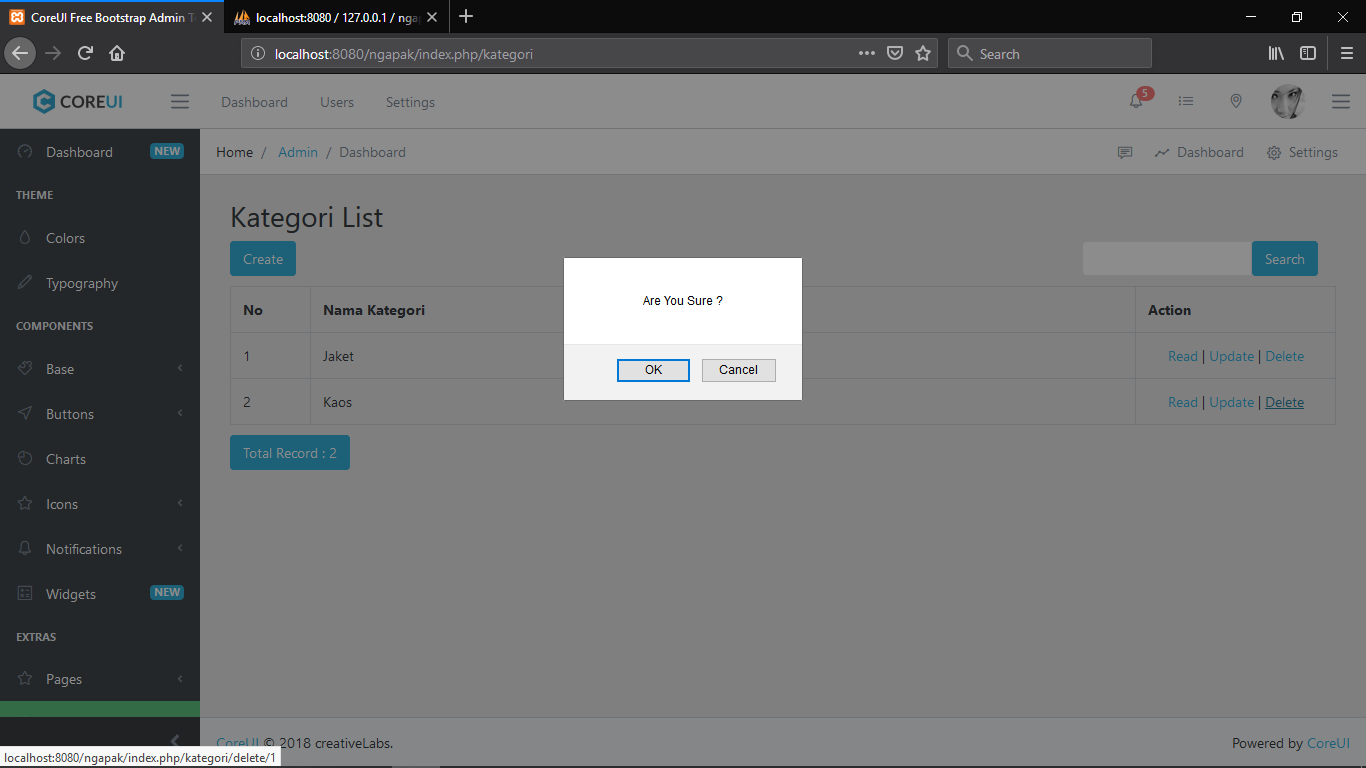
**Gambar 3.71 Halaman *Create* Kategori Admin**

* Halaman *Update* KategoriAdmin



**Gambar 3.72 Halaman *Update* Kategori Admin**

* Halaman *Delete* KategoriAdmin



**Gambar 3.73 Halaman *Delete* Kategori Admin**

**4.2 Pembahasan**

**4.2.1 Kode Program *Model***

Dalam kode program *model* yang ditampilakan berikut ini adalah *model* kategori.

|  |
| --- |
| public $table = 'kategori';  public $id = 'id\_kategori';  public $order = 'DESC';  function \_\_construct()  {  parent::\_\_construct();  }  // get all  function get\_all()  {  $this->db->order\_by($this->id, $this->order);  return $this->db->get($this->table)->result();  }  // get data by id  function get\_by\_id($id)  {  $this->db->where($this->id, $id);  return $this->db->get($this->table)->row();  }    // get total rows  function total\_rows($q = NULL) {  $this->db->like('id\_kategori', $q);  $this->db->or\_like('nama\_kategori', $q);  $this->db->from($this->table);  return $this->db->count\_all\_results();  } |

Dari script coding diatas, dijelaskan bahwa yang pertama dilakukan adalah mendklarasikan variabel yang kita butuhkan dalam *model* kategori. Kemudian menggunakan method panggil semua isi dari kategori, lalu diurutkan secara descending. Link untuk *codingan* diatas :

<https://github.com/adhysyahputraaa/RPL-16S1SI06-11-TokoSablonKaos/blob/master/ngapak/application/models/Kategori_model.php>

**4.2.2 Kode Program *View***

Dalam kode program *view* yang ditampilakan berikut ini adalah *view* kategori. *View* yang ditampilkan dibawah ini adalah *list* dan *form*.

**Tabel 4.2 *Script Code View List* Kategori**

|  |
| --- |
| <th>No</th>  <th>Nama Kategori</th>  <th>Action</th>  </tr><?php  foreach ($kategori\_data as $kategori)  {  ?>  <tr>  <td width="80px"><?php echo ++$start ?></td>  <td><?php echo $kategori->nama\_kategori ?></td>  <td style="text-align:center" width="200px">  <?php  echo anchor(site\_url('kategori/read/'.$kategori->id\_kategori),'Read');  echo ' | ';  echo anchor(site\_url('kategori/update/'.$kategori->id\_kategori),'Update');  echo ' | ';  echo anchor(site\_url('kategori/delete/'.$kategori->id\_kategori),'Delete','onclick="javasciprt: return confirm(\'Are You Sure ?\')"');  ?>  </td>  </tr> |

Dari script coding diatas, dijelaskan bahwa memiliki *tabel header* bertuliskan no, nama kategori, dan *action* kemudian menampilankan data nama\_kategori dengan cara menggunakan variabel $kategori. Kemudian dalam setiap baris dari data akan memiliki aksi *read*, *update*, dan *delete*. Setiap aksi memiliki url yang berbeda-beda. Link untuk *codingan* diatas :

<https://github.com/adhysyahputraaa/RPL-16S1SI06-11-TokoSablonKaos/blob/master/ngapak/application/views/kategori/kategori_list.php>

**Tabel 4.2 *Script Code View Form* Kategori**

|  |
| --- |
| <?php  $this->load->view('template/head');  $this->load->view('template/header');  $this->load->view('template/nav');  ?>  <h2 style="margin-top:0px">Kategori <?php echo $button ?></h2>  <form action="<?php echo $action; ?>" method="post">  <div class="form-group">  <label for="varchar">Nama Kategori <?php echo form\_error('nama\_kategori') ?></label>  <input type="text" class="form-control" name="nama\_kategori" id="nama\_kategori" placeholder="Nama Kategori" value="<?php echo $nama\_kategori; ?>" />  </div>  <input type="hidden" name="id\_kategori" value="<?php echo $id\_kategori; ?>" />  <button type="submit" class="btn btn-primary"><?php echo $button ?></button>  <a href="<?php echo site\_url('kategori') ?>" class="btn btn-default">Cancel</a>  </form>  <?php $this->load->view('template/footer'); ?> |

Dari script coding diatas, dijelaskan bahwa akan memuat sebuah head, header, nav, dan footer yang berada di view dalam folder template kemudian menampilkan judul “Kategori $button”, yang artinya $button akan diisi dengan apa yang telah pengguna pilih, jika pengguna memilih update makan akan menampilkan *update* dan *create* juga akan menampilkan *create*. Setelah itu terdapat label dengan tulisan Nama Kategori yang bersebelahan dengan *form* input bertipe *text*. Jika pengguna memilih *update* maka akan memilik nilai disetiap *form* inputannya. Id\_kategori disini di-*hidden* atau disembunyikan. Dibawah dari form inputan terdapat sebuah *button* yang kemudian jika diklik akan diarahkan ke halaman *list* kategori kembali. Link untuk *codingan* diatas :

<https://github.com/adhysyahputraaa/RPL-16S1SI06-11-TokoSablonKaos/blob/master/ngapak/application/views/kategori/kategori_form.php>

**4.2.3 Kode Program *Controller***

Dalam kode program *controller* yang ditampilakan berikut ini adalah *controller* kategori. *Method* yang ditampilkan dibawah ini adalah *update* dan *delete*. Link untuk *codingan* dibawah :

<https://github.com/adhysyahputraaa/RPL-16S1SI06-11-TokoSablonKaos/blob/master/ngapak/application/controllers/Kategori.php>

**Tabel 4.1 *Script Code Method Update* Kategori**

|  |
| --- |
| public function update($id)  {  $row = $this->Kategori\_model->get\_by\_id($id);  if ($row) {  $data = array(  'button' => 'Update',  'action' => site\_url('kategori/update\_action'),  'id\_kategori' => set\_value('id\_kategori', $row->id\_kategori),  'nama\_kategori' => set\_value('nama\_kategori', $row->nama\_kategori),  );  $this->load->view('kategori/kategori\_form', $data);  } else {  $this->session->set\_flashdata('message', 'Record Not Found');  redirect(site\_url('kategori'));  }  } |

Dari *script coding* diatas, dijelaskan bahwa pertama akan dilakukan proses *update* sesuai dengan id yang telah dipilih. Kemudian jika *button update* dipilih maka akan menampilkan sebuah form data *update* kategori yang di dalamnya sudah terdapat nilai dari id\_kategori dan nama\_kategori yang sebelumnya sudah terisi. Kemudian jika tidak memilik data akan ditampilkan sebuah pesan yaitu “*Record Not Found*”.

**Tabel 4.2 *Script Code Method Delete* Kategori**

|  |
| --- |
| public function delete($id)  {  $row = $this->Kategori\_model->get\_by\_id($id);  if ($row) {  $this->Kategori\_model->delete($id);  $this->session->set\_flashdata('message', 'Delete Record Success');  redirect(site\_url('kategori'));  } else {  $this->session->set\_flashdata('message', 'Record Not Found');  redirect(site\_url('kategori'));  }  } |

Dari *script coding* diatas, dijelaskan bahwa pertama akan dilakukan proses *update* sesuai dengan id yang telah dipilih. Kemudian jika *button delete* dipilih maka akan menghapus data tersebut. Kemudian jika tidak memilik data akan ditampilkan sebuah pesan yaitu “*Record Not Found*”.

**4.2.4 Presentasi Hasil**

Berikut ini adalah link youtube untuk presentasi hasil : <https://www.youtube.com/watch?v=i-JCwafYJTc&t=3s>