


**LAPORAN PROYEK**  
**KOM 205/ Basis Data**  
**DripShip**  
**Expedition For Your Fashion Cargo**

Dipersiapkan oleh:

<b>Patar Isac Pardomuan</b>	<b>G6401201092</b>
<b>Daffa Rifqi Kanz</b>	<b>G6401201094</b>
<b>Pulung Rafi' Muhammad</b>	<b>G6401201089</b>
<b>Dzakiyyah Hasbi Hutaurok</b>	<b>G6401201006</b>

Departemen Ilmu Komputer  
Fakultas Matematika dan Ilmu Pengetahuan Alam  
Institut Pertanian Bogor  
2021

	<b>Departemen Ilmu Komputer</b> <b>Fakultas Matematika dan IPA</b> <b>Institut Pertanian Bogor</b>	<b>Nomor Dokumen</b>		<b>Halaman</b>
		<i>BASDAT – 110</i>		<i>1/16</i>
		<i>Revisi</i>		<i>Tgl: 20/12/21</i>

## Daftar Isi

1 Ringkasan.....	3
2 Pendahuluan.....	3
2.1 Latar Belakang .....	3
2.2 Rumusan Masalah .....	3
2.3 Tujuan.....	3
2.4 Solusi Singkat.....	4
3 Skema Basis Data .....	5
3.1 Entity-Relational Diagram .....	5
3.2 Diagram Skematik.....	6
4 Implementasi.....	7
4.1 Proses Implementasi.....	7
4.2 Hasil Implementasi.....	12
4.2.1 Home.....	12
4.2.2 Shipment .....	13
4.2.3 Details .....	13
5 Pembagian Kerja dalam Kelompok .....	14
6 Lampiran .....	14
6.1 Log Activity Anggota Kelompok.....	14
6.1.1 Aktivitas GitHub.....	16

# 1 Ringkasan

Permasalahan kurir sudah menjadi hal yang cukup genting dengan pertumbuhan E-Commerce yang sangat cepat. Sudah banyak cerita di media sosial dimana paket yang diterima oleh konsumen tidak sesuai dengan apa yang dibeli. Ditambah dengan penanganan paket yang tidak sesuai dengan barang yang akan dikirim, terutama barang *fashion* yang sensitif. Diperlukan sebuah solusi untuk menjadi ekspedisi yang fokus dalam pengiriman paket barang *fashion*.

Dalam implementasi aplikasi, kami memanfaatkan PostgreSQL sebagai DBMS, html sebagai front-end, dan php sebagai back-end. Dalam proses implementasi *conceptual diagram* menjadi sebuah website, kami memanfaatkan bantuan *bootstrap* versi 5 untuk front-end. Sebelum itu, kami membuat ERD dan kami normalisasi diagram tersebut agar cocok dan sesuai dengan apa yang kami butuhkan untuk membuat, membaca, menampilkan, dan mengolah data (CRUD).

Website DripShip berfungsi sebagai alat administrasi *order* pengiriman paket dengan fitur utama CRUD sehingga data dapat dimodifikasi secara *real-time*. Website berhasil jalan dengan baik dan sesuai rencana pembuatan. Data yang diuji coba tersimpan dengan baik dan dapat dimodifikasi secara langsung dari website sesuai desain awal.

## 2 Pendahuluan

### 2.1 Latar Belakang

*E-Commerce* sudah menjadi pasar terbesar di dunia. Pasar ini tidak memiliki wujud namun tetap menawarkan keamanan transaksi dalam bentuk bermacam-macam. Untuk menyelesaikan transaksi maka diperlukan orang ketiga (kurir) untuk mengantarkan barang jual ke customer yang telah membeli barang tersebut. Biasanya perusahaan penyedia layanan kurir disebut Ekspedisi.

Sedang dimasa ini juga sudah merajalela oknum-oknum yang membongkar paket customer atau merusak paket dengan sengaja ataupun tidak disengaja. Tanpa terkecuali barang jual *fashion*. Barang ini mudah rusak karena bahannya yang mudah rusak atau karena pengemasan yang kurang kuat. Oleh karena itu, diperlukan sebuah ekspedisi yang akan menjadi solusi masalah ini. Hadirlah DripShip.

### 2.2 Rumusan Masalah

1. Bagaimana kualitas penanganan paket barang *fashion*?
2. Apakah ada ekspedisi yang mengkhususkan diri untuk mengirim barang *fashion*?
3. Apa solusi untuk masalah pengiriman barang *fashion*?

### 2.3 Tujuan

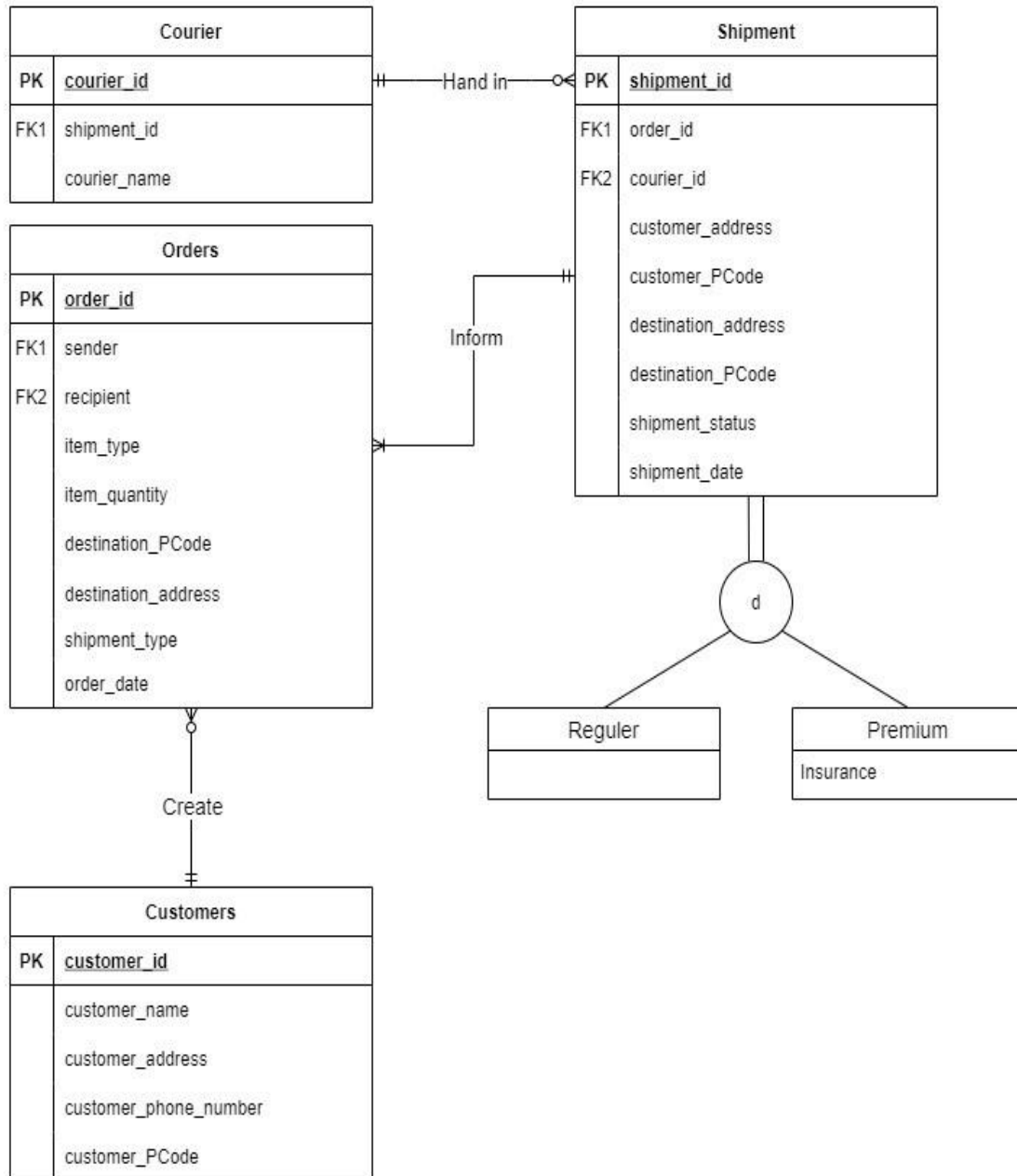
Aplikasi ini dibuat sebagai solusi permasalahan mengenai penanganan pengiriman barang *fashion* yang tidak sesuai dengan baik. Juga untuk menuntaskan tugas akhir mata kuliah Basis Data KOM20F.

## 2.4 Solusi Singkat

DripShip hadir sebagai solusi dalam pengiriman paket barang *fashion*. Ekspedisi ini akan menangani pengiriman sesuai dengan jenis barang *fashion* yang akan dikirim. DripShip juga menawarkan pengiriman premium yang akan mendapatkan asuransi untuk paket yang akan dikirim. Dengan begitu paket akan sampai dalam keadaan aman dan utuh dan penerima bisa tenang menunggu paket sampai.

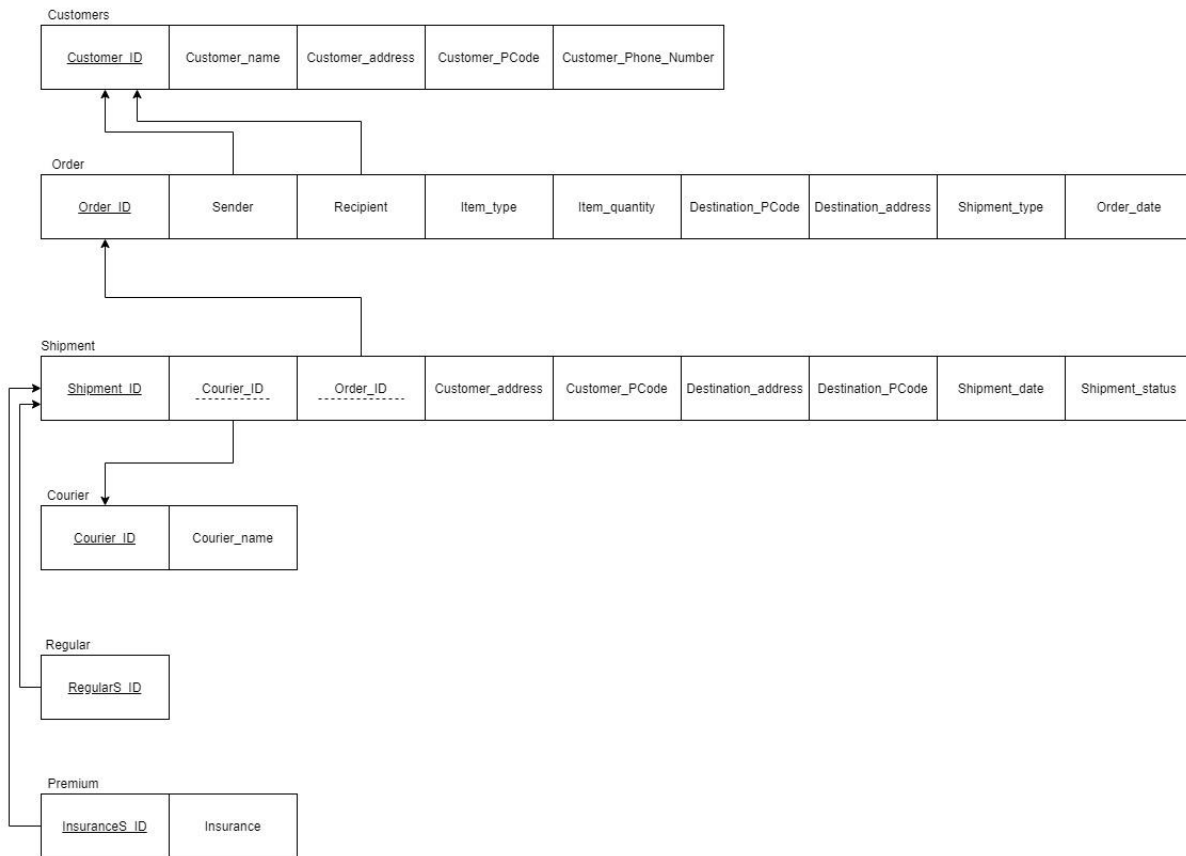
### 3 Skema Basis Data

#### 3.1 Entity-Relational Diagram



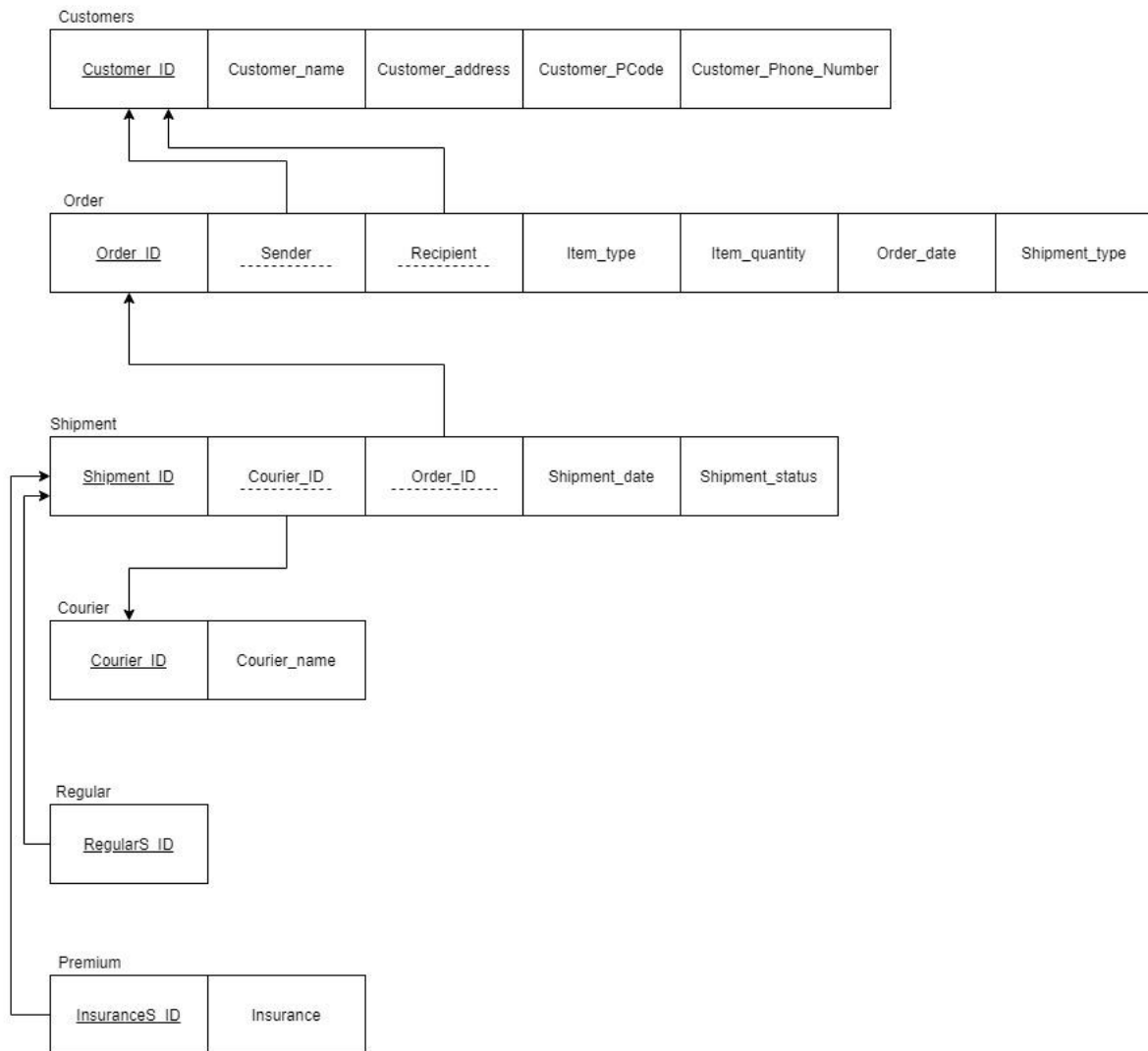
## 3.2 Diagram Skematik

### UNF



\*Pcode = Postal Code

### 3NF



\*Pcode = Postal Code

## 4 Implementasi

### 4.1 Proses Implementasi

DripShip diimplementasikan dengan membuat aplikasi berbentuk website yang diisi dengan formulir pesanan dan tabel status pengiriman. Pembuatan *front-end* website dilakukan menggunakan bahasa *html* dengan bantuan source *bootstrap* versi 5. *Bootstrap* memudahkan pembuatan website sehingga website terlihat lebih menarik. Setelah website sudah terbentuk, *html* diubah menjadi *php* untuk memasukkan basis data kedalam website.

Hal pertama yang harus disiapkan sebelum mengembangkan *back-end* adalah membuat *config.php* yang berguna untuk menghubungkan web dengan database. Di *config.php* ini kita menggunakan `pg_connect()` karena DBMS yang kami gunakan adalah PostgreSQL. Berikut adalah potongan kode dari *config.php*.

```

1 <?php
2 $db=pg_connect('host=localhost dbname=dripship user=postgres password=patar');
3 if( !$db ){
4     die("Gagal terhubung dengan database: " . pg_connect_error());
5 }
6 ?>

```

File *config.php* ini akan banyak di-include di file php lainnya. Untuk proses CRUD, *create* dimulai dari *index.php*, halaman web utama dari DripShip yang fungsi utamanya adalah untuk menambah order. Di halaman *index* ini tersedia form untuk data pengirim (*sender*), penerima (*recipient*), dan detail order, yang nantinya akan disimpan di DBMS. Value dari form ini selanjutnya akan dikirim ke *add\_order.php* dengan method POST. Di *add\_order.php* ini, semua value yang diterima dengan method POST akan dimuat ulang dan dimasukkan ke dalam variable agar lebih rapi, kemudian dilakukan query untuk menyimpannya ke DBMS. Disini ada empat proses query, yang pertama adalah menyimpan data pengirim ke tabel customers, lalu menyimpan data penerima ke tabel customers juga, lalu menambah data order ke tabel orders, dan yang terakhir secara otomatis menambah data pengiriman (*shipment*) ke tabel shipment. Berikut adalah potongan kode dari *add\_order.php*

```

. . .
$s_name = $_POST['s-name'];
$s_addr = $_POST['s-address'];
$s_pcode = $_POST['s-pcode'];
$s_phone = $_POST['s-phone'];
$r_name = $_POST['r-name'];
$r_addr = $_POST['r-address'];
$r_pcode = $_POST['r-pcode'];
$r_phone = $_POST['r-phone'];
$item_type = $_POST['item-type'];
$item_qty = $_POST['item-qty'];
$ship_type = $_POST['ship-type'];
$ttl = date('d');
$bln = date('M');
$thn = date('Y');

$query_s = pg_query("INSERT INTO customers (name,address,pcode,phone_number) VALUES('$s_name','$s_addr', '$s_pcode', '$s_phone')");
$s_id = pg_fetch_result(pg_query("SELECT count(*) from customers"), "count");
$query_r = pg_query("INSERT INTO customers (name,address,pcode,phone_number) VALUES('$r_name','$r_addr', '$r_pcode', '$r_phone')");
$r_id = pg_fetch_result(pg_query("SELECT count(*) from customers"), "count");
$query_o = pg_query("INSERT INTO orders (sender_id,recipient_id,item_type,item_qty,order_date,shipment_type) VALUES($s_id,$r_id,'$item_type','$item_qty','$ttl-$bln-$thn','$ship_type')");
$o_id = pg_fetch_result(pg_query("SELECT count(*) from orders"), "count");
$query_sh = pg_query("SELECT add_ship($o_id,'$ttl-$bln-$thn')");
. . .

```



Proses CRUD selanjutnya adalah *read*, terdapat di halaman *shipment.php*. Di sana dilakukan query untuk tabel shipment sudah dibagi menjadi dua view, yaitu *ongoing* dan *done*. View *ongoing* adalah untuk shipment yang masih dalam proses pengiriman, view *done* adalah untuk shipment yang sudah selesai. Data-data hasil query tadi ditampilkan dalam bentuk tabel. Berikut adalah potongan kode dari proses query di *shipment.php*.

```
<?php
$query = pg_query("SELECT * FROM done");
while($ship = pg_fetch_array($query)){
    echo "<tr>";
    echo "<td>". "SH-". $ship["id"]. "</td>";
    echo "<td>". $ship["shipment_date"]. "</td>";
    echo "<td>". $ship["shipment_type"]. "</td>";
    echo "<td>". "CR-". $ship["courier_id"]. "</td>";
    echo "<td>". $ship["shipment_status"]. "</td>";
    echo "<td><a href='\"details.php?id=".$ship["id"]."\"' class='\"btn btn-info btn-sm\"'>Details</a> <a href='\"edit.php?id=".$ship["id"]."\"' class='\"btn btn-secondary btn-sm\"'>Edit</a> <a href='\"delete.php?id=".$ship["id"]."\"' class='\"btn btn-danger btn-sm\"'>Delete</a></td>";
    echo "</tr>";
}
?>
```

Selain di *shipment.php*, proses *read* juga dilakukan di *details.php*. Di sana dilakukan pembacaan semua data dari semua tabel untuk satu order atau pengiriman. Berikut adalah potongan query dari *details.php*.

```
1 <?php
2 include("config.php");
3 $id = $_GET["id"];
4 $q_ship = pg_query("SELECT * FROM shipment WHERE id=$id");
5 $ship = pg_fetch_array($q_ship);
6 $order_id = $ship["order_id"];
7 $q_order = pg_query("SELECT * FROM orders WHERE id=$order_id");
8 $order = pg_fetch_array($q_order);
9 $sender_id = $order["sender_id"];
10 $recip_id = $order["recipient_id"];
11 $item_type = $order["item_type"];
12 $item_qty = $order["item_qty"];
13 $order_date = $order["order_date"];
14 $ship_type = $order["shipment_type"];
15 $ship_status = $ship["shipment_status"];
16 $ship_date = $ship["shipment_date"];
17 $courier_id = $ship["courier_id"];
18 $q_sender = pg_query("SELECT * FROM customers WHERE id=$sender_id");
19 $sender = pg_fetch_array($q_sender);
20 $sender_name = $sender["name"];
21 $sender_addr = $sender["address"];
22 $sender_pcode = $sender["pcode"];
23 $sender_pnumber = $sender["phone_number"];
24 $q_recip = pg_query("SELECT * FROM customers WHERE id=$recip_id");
25 $recip = pg_fetch_array($q_recip);
26 $recip_name = $recip["name"];
27 $recip_addr = $recip["address"];
28 $recip_pcode = $recip["pcode"];
29 $recip_pnumber = $recip["phone_number"];
30 $q_courier = pg_query("SELECT * FROM courier WHERE id=$courier_id");
31 $courier = pg_fetch_array($q_courier);
32 $courier_name = $courier["name"];
33
```

Proses CRUD selanjutnya adalah *update*, terdapat pada proses update status pengiriman di *edit.php* dan *process\_edit.php*. Di *edit.php* hanya terdapat form select yang value-nya adalah “In shipping” dan “Shipped”. Lalu data ini dikirim ke *process\_edit.php* bersamaan dengan id dari shipment dengan metode GET. Di *process\_edit.php* dilakukan proses untuk update data status pengiriman di tabel shipment. Berikut adalah potongan kode *process\_edit.php*

```

1  <?php include("config.php");
2
3  if(isset($_POST['status'])){
4
5      $id = $_POST["id"];
6      $status = $_POST["status"];
7
8
9      // update query
10     $query = pg_query("UPDATE shipment SET shipment_status = '$status' WHERE id = $id");
11
12     if( $query==TRUE ) {
13         header('Location: shipment.php');
14     } else {
15         die("GAGAL GAN");
16     }
17
18
19 } else {
20     die("GAGAL GAN");
21 }
22
23 ?>

```

Untuk proses CRUD yang terakhir, *delete*, terdapat pada *delete.php* yang fungsinya adalah untuk menghapus data dari tabel *shipment* saja. Berikut adalah potongan kode dari *delete.php*

```

1  <?php include("config.php");
2  $id = $_GET["id"];
3
4  $query = pg_query("DELETE FROM shipment WHERE id = $id");
5  if( $query==TRUE ) {
6      // ubah berhasil alihkan ke daftarsiswa.php
7      header('Location: shipment.php');
8  } else {
9      die("gagal mengubah..");
10 }
11 ?>

```

Untuk proses implementasi database, seperti yang sudah dijelaskan di atas, kami menggunakan PostgreSQL versi 13. Tabel-tabel yang dibuat adalah sama seperti tabel-tabel yang sudah dinormalisasi, kecuali untuk tipe pengiriman (*shipment\_type*), kami menambahkannya sebagai kolom di tabel order. Berikut adalah command SQL yang digunakan untuk membuat tabel-tabel yang dibutuhkan.

```

1 CREATE TABLE customers(
2   id serial,
3   name VARCHAR(64) NOT NULL,
4   address VARCHAR(200) NOT NULL,
5   pcode VARCHAR(10) NOT NULL,
6   phone_number VARCHAR(13) NOT NULL,
7   CONSTRAINT customers_id_PK PRIMARY KEY(id)
8 );
9
10 CREATE TABLE orders(
11   id serial,
12   sender_id INT NOT NULL,
13   recipient_id INT NOT NULL,
14   item_type VARCHAR(20) NOT NULL,
15   item_qty INT NOT NULL,
16   order_date DATE NOT NULL,
17   shipment_type VARCHAR(20) NOT NULL,
18   CONSTRAINT order_sender_FK FOREIGN KEY(sender_id) REFERENCES customers(id),
19   CONSTRAINT order_recipient_FK FOREIGN KEY(recipient_id) REFERENCES customers(id),
20   CONSTRAINT orders_id_PK PRIMARY KEY(id)
21 );
22
23 CREATE TABLE courier(
24   id serial,
25   name VARCHAR(64) NOT NULL,
26   CONSTRAINT courier_id_PK PRIMARY KEY(id)
27 );
28
29 CREATE TABLE shipment(
30   id serial,
31   courier_id INT NOT NULL,
32   order_id INT NOT NULL,
33   shipment_date DATE NOT NULL,
34   shipment_status VARCHAR(20) NOT NULL,
35   CONSTRAINT shipment_courier_id_FK FOREIGN KEY(courier_id) REFERENCES courier(id),
36   CONSTRAINT shipment_order_id_FK FOREIGN KEY(order_id) REFERENCES orders(id)
37 );

```

Lalu agar lebih mudah, kami membuat sebuah function yang digunakan pada *add\_order.php* yang berguna untuk menambah shipment secara otomatis dengan merandom kurir & menambahkan tanggal pengiriman, yaitu 3 hari setelah tanggal order. Berikut adalah potongan command SQL nya.

```

39  /* untuk otomatis input shipment ketika create orders di website */
40  CREATE OR REPLACE FUNCTION add_ship(o_id INT, o_date DATE)
41  RETURNS integer AS $$
42  BEGIN
43    INSERT INTO shipment (courier_id, order_id, shipment_date, shipment_status)
44    VALUES(floor(random()* (5-1 + 1) + 1),o_id,o_date + 3,'In shipping');
45    RETURN 1;
46  END;
47  $$ LANGUAGE plpgsql;

```

Lalu, kami juga membuat *view* untuk shipment yang sedang dalam proses pengiriman dan *view* untuk shipment yang sudah selesai. View ini dibuat untuk memudahkan proses query di *shipment.html* karena di sana kami memisahkan tabel antara pengiriman yang sudah selesai dan yang masih dalam pengiriman. Berikut adalah command SQL untuk membuat view-nya.

```

49  /* create view untuk display di website */
50  /* on going alias belum dikirim */
51  CREATE OR REPLACE VIEW ongoing AS
52  SELECT S.order_id, S.id, S.shipment_date, O.shipment_type, S.courier_id, S.shipment_status
53  FROM shipment S, orders O
54  WHERE S.order_id = O.id AND S.shipment_status = 'In shipping'
55  ORDER BY S.id;
56  /* done alias sudah dikirim */
57  CREATE OR REPLACE VIEW done AS
58  SELECT S.order_id, S.id, S.shipment_date, O.shipment_type, S.courier_id, S.shipment_status
59  FROM shipment S, orders O
60  WHERE S.order_id = O.id AND S.shipment_status = 'Shipped'
61  ORDER BY S.id;

```

## 4.2 Hasil Implementasi

### 4.2.1 Home

DripShip! Home Create Order Shipment

### Customer Data

Sender	Recipient
Name <input type="text" value="Default input"/>	Name <input type="text" value="Default input"/>
Address <input type="text" value="Default input"/>	Address <input type="text" value="Default input"/>
Postal Code <input type="text" value="Default input"/>	Postal Code <input type="text" value="Default input"/>
Phone Number <input type="text" value="Default input"/>	Phone Number <input type="text" value="Default input"/>

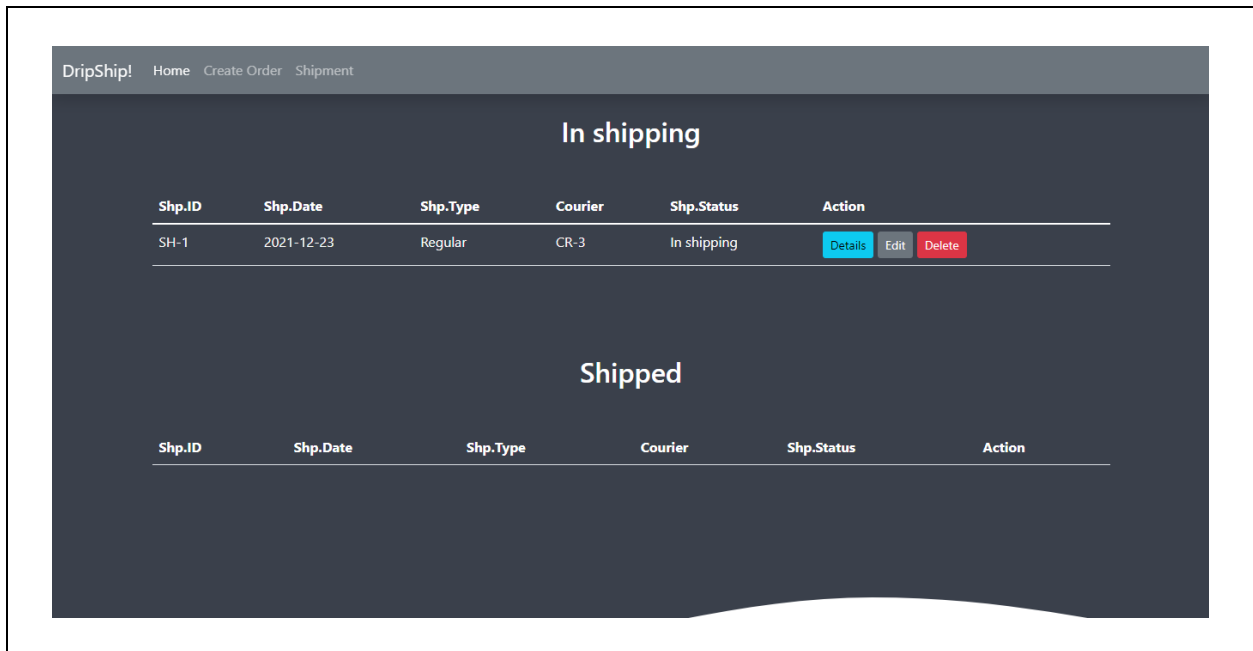
### Order Data

Item Type

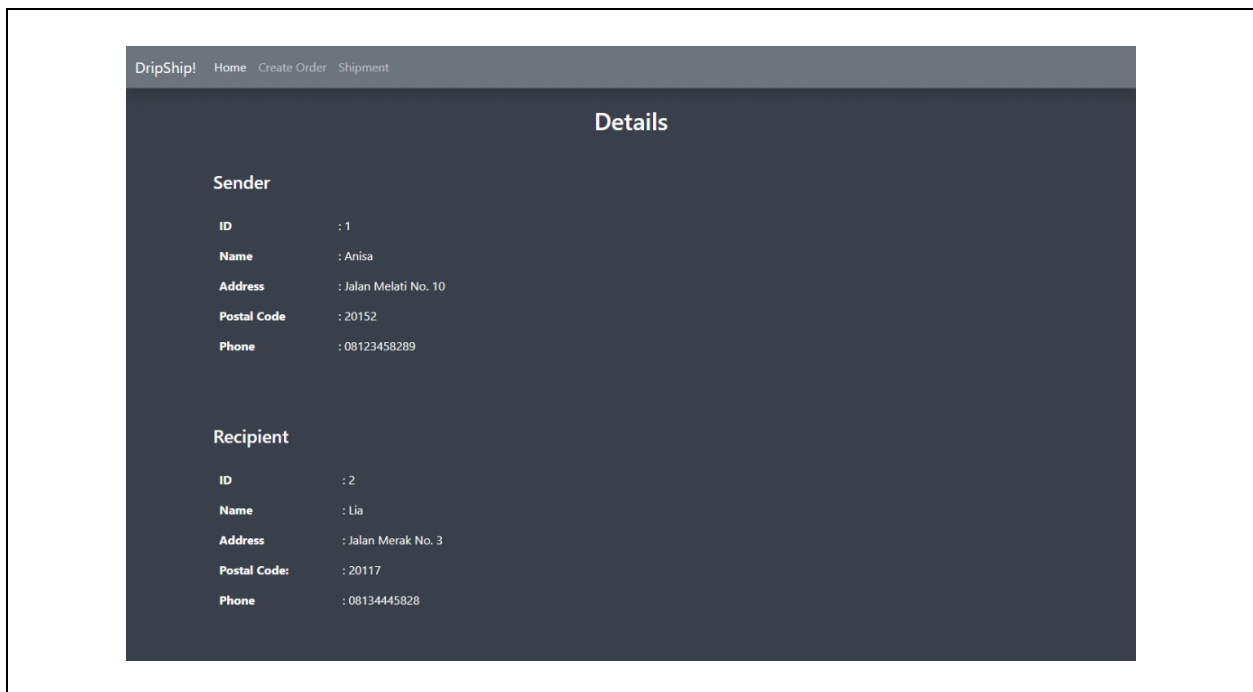
Item Quantity

Shipment Type  
Regular ▼

## 4.2.2 Shipment



## 4.2.3 Details



Order

Order ID

: 1

Order Date

: 2021-12-20

Item Type

: Kalung

Item Quantity

: 3

Shipment

Shipment ID

: 1

Shipment Date

: 2021-12-23

Shipment Type

: Regular

Courier ID

: 3

Courier Name

: Gilang

Shipment Status

: In shipping

Back

## 5 Pembagian Kerja dalam Kelompok

Nama	Tugas
Patar Isac Pardomuan	Back-end dan DBMS
Daffa Rifqi Kanz	Normalisasi, Dokumentasi
Pulung Rafi' Muhammad	Front-end, ERD
Dzakiyyah Hasbi Hutaeruk	EERD, Normalisasi, Front-end

## 6 Lampiran

### 6.1 Log Activity Anggota Kelompok

Task	PIC	Timeline													
		Minggu - 1							Minggu - 2						
Documentation		1	2	3	4	5	6	7	1	2	3	4	5	6	7
Laporan	Semua anggota														
Video	Daffa RK														
Presentation	Daffa RK														
Diagrams															

ERD	Semua anggota
Relational Table	Dzakiyyah
Normalization	Daffa RK
Back-end	
Database	Patar
Home	Patar
Shipment	Patar
Edit	Patar
Details	Patar
Delete	Patar
Front-end	
Home	Pulung R / Dzakiyyah
Shipment	Pulung R / Dzakiyyah
Edit	Pulung R
Details	Pulung R
Readme	Dzakiyyah

## 6.1.1 Aktivitas GitHub

