**LAPORAN PRAKTIKUM PEMROGRAMAN INTERNET**

****

**Oleh**

**KELOMPOK 20**

**Chesia Christy Natalita Br Tarigan (1905551086)**

**Nyoman Wisnu Laksana (1905551128)**

**Gede Jaya Bhaswara (1905551129)**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS UDAYANA**

**2021**

# 

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan berkat- Nya sehingga pengerjaan Laporan Praktikum Pemrograman Internet ini dapat berjalan dengan baik. Laporan Praktikum Pemrograman Internet ini disusun dengan tujuan untuk memenuhi tugas mata kuliah Praktikum Pemrograman Internet dalam menempuh pendidikan di Jurusan Teknologi Informasi, Fakultas Teknik, Universitas Udayana.

Selama pembuatan Laporan Praktikum Pemrograman Internet, terdapat banyak pihak yang terlibat dalam penyempurnaannya, untuk itu pada kesempatan ini disampaikan terima kasih kepada pihak-pihak yang terlibat antara lain.

1. I Made Sunia Raharja, S.Kom., M.Cs selaku dosen pengampu.
2. Asisten dosen yang telah membimbing dalam melakukan praktikum dan penyusunan Laporan Praktikum Pemrograman Internet ini.
3. Teman-teman dan semua pihak yang telah berjasa memberikan motivasi dalam proses menyelesaikan Laporan Praktikum Pemrograman Internet ini.

Menyadari bahwa masih banyak kekurangan dan keterbatasan dalam penyajian data dalam Laporan Praktikum Pemrograman Internet ini, saran serta kritik yang membangun diperlukan untuk perbaikan ke depannya. Demikian Laporan Praktikum Pemrograman Internet ini disusun, apabila ada kesalahan penulisan kata yang disengaja atau tidak mohon dimaafkan.

Bukit Jimbaran, April 2021

Tim Penyusun

# ABSTRAK

Teknologi informasi berkembang dengan sangat pesat di dunia yang serba digital saat ini. Melalui perkembangan teknologi informasi, efisiensi dari penyimpanan informasi semakin meningkat dan sistem informasi terstruktur sudah mulai banyak diterapkan oleh organisasi-organisasi pada berbagai bidang. *Website* merupakan kumpulan halaman untuk menampilkan suatu informasi berupa teks, gambar, animasi, suara, dan lainnya. *Website* bukan lagi barang langka dan “mahal” karena sudah sangat sering dipakai bukan hanya oleh organisasi handal melainkan juga individu yang ingin menggunakan sistem informasi dan penyimpanan informasi saat ini. Salah satu penggunaan *website* yang paling sering digunakan yaitu untuk keperluan bisnis. Oleh karena itu, muncul sebuah istilah *e-commerce* atau suatu situs internet transaksi perdagangan yang mempertemukan penjual dan pembeli dengan media internet. Dengan adanya *e-commerce*, pebisnis-pebisnis baru dapat mengembangkan bisnisnya tanpa harus menyewa ruko untuk transaksi jual beli. Pebisnis hanya perlu membekali diri dengan internet. Begitu juga dengan pebisnis handal yang dapat melebarkan cakupan pelanggannya tanpa batasan wilayah. Penerapan *e-commerce* cukup beragam tergantung dari kebutuhan, namun secara umum *software* yang biasa digunakan untuk membuat sebuah *website* yaitu aplikasi pengelolaan *database* seperti MySQL serta bahasa pemrograman internet seperti HTML, PHP, maupun Javascript.

**Kata Kunci:** *E-commerce*, *PHP*, *MySQL*, *HTML*

# DAFTAR ISI

[KATA PENGANTAR ii](#_Toc74351985)

[ABSTRAK iii](#_Toc74351986)

[DAFTAR ISI iv](#_Toc74351987)

[DAFTAR GAMBAR vi](#_Toc74351988)

[DAFTAR TABEL viii](#_Toc74351989)

[DAFTAR KODE PROGRAM ix](#_Toc74351990)

[BAB I PENDAHULUAN 1](#_Toc74351991)

[1.1 Latar Belakang 1](#_Toc74351992)

[1.2 Rumusan Masalah 2](#_Toc74351993)

[1.3 Tujuan 2](#_Toc74351994)

[1.4 Manfaat 2](#_Toc74351995)

[1.5 Batasan Masalah 3](#_Toc74351996)

[1.6 Sistematika Penulisan 3](#_Toc74351997)

[BAB II TINJAUAN PUSTAKA 5](#_Toc74351998)

[2.1 Pemrograman Internet 5](#_Toc74351999)

[2.2 HTML 5](#_Toc74352000)

[2.2.1 Tag Dasar HTML 6](#_Toc74352001)

[2.3 PHP 10](#_Toc74352002)

[2.3.1 Laravel 11](#_Toc74352003)

[2.4 CSS 14](#_Toc74352004)

[2.5 JavaScript 14](#_Toc74352005)

[2.6 JQuery 15](#_Toc74352006)

[2.7 MySQL 16](#_Toc74352007)

[2.8 Git dan Github 17](#_Toc74352008)

[2.8.1 Repositori 17](#_Toc74352009)

[2.8.2 Branch 18](#_Toc74352010)

[2.8.3 Pull Request 18](#_Toc74352011)

[2.8.4 Forking Repositori 18](#_Toc74352012)

[BAB III METODE PENELITIAN 19](#_Toc74352013)

[3.1 Tempat dan Waktu Penelitian 19](#_Toc74352014)

[3.2 Gambaran Umum Sistem 19](#_Toc74352015)

[3.3 Perancangan Database 20](#_Toc74352016)

[3.3.1 PDM 21](#_Toc74352017)

[3.3.2 Kamus Data 22](#_Toc74352018)

[BAB IV PEMBAHASAN DAN HASIL ANALISA 40](#_Toc74352019)

[4.1 Multi Authentication 40](#_Toc74352020)

[4.1.1 Login Admin 43](#_Toc74352021)

[4.1.2 Login User 49](#_Toc74352022)

[4.1.3 Register User 57](#_Toc74352023)

[4.2 Master dan Detail Data 68](#_Toc74352024)

[4.2.1 CRUD pada Tabel Product 68](#_Toc74352025)

[4.2.2 CRUD pada Tabel Courier 82](#_Toc74352026)

[4.2.3 CRUD pada Tabel Category 93](#_Toc74352027)

[4.3 Transaksi 102](#_Toc74352028)

[4.3.1 Add to Chart 102](#_Toc74352029)

[4.3.2 Membatalkan transaksi 127](#_Toc74352030)

[4.3.4 Upload Bukti Pembayaran 132](#_Toc74352031)

[4.3.5 Status Transaksi 137](#_Toc74352032)

[4.3.6 Review dan Response 145](#_Toc74352033)

[4.4 Report 153](#_Toc74352034)

[4.4.1 Notifikasi Pembeli 153](#_Toc74352035)

[4.4.2 Notifikasi Admin 157](#_Toc74352036)

[4.4.3 Laporan Transaksi 161](#_Toc74352037)

[BAB V PENUTUP 170](#_Toc74352038)

[5.1 Kesimpulan 170](#_Toc74352039)

[5.2 Saran 170](#_Toc74352040)

[DAFTAR PUSTAKA 171](#_Toc74352041)

# DAFTAR GAMBAR

[Gambar 3. 1 Gambaran Umum Sistem 26](#_Toc74350104)

[Gambar 3. 2 Rancangan PDM *E-Commerce* Berbasis Web 27](#_Toc74350105)

[Gambar 4. 1 Tampilan Halaman *Login* 55](#_Toc74350113)

[Gambar 4. 2 Tampilan Halaman *Admin* 56](#_Toc74350114)

[Gambar 4. 3 Tampilan Halaman *Login User* 63](#_Toc74350115)

[Gambar 4. 4 Tampilan Halaman Utama 63](#_Toc74350116)

[Gambar 4. 5 Tampilan Halaman *Register User* 73](#_Toc74350117)

[Gambar 4. 6 Tampilan Halaman *Login User* 74](#_Toc74350118)

[Gambar 4. 7 Tampilan Halaman Utama 74](#_Toc74350119)

[Gambar 4. 8 Halaman CRUD Tabel *Product* 85](#_Toc74350120)

[Gambar 4. 9 Halaman *Create* 86](#_Toc74350121)

[Gambar 4. 10 Halaman CRUD *Product* 86](#_Toc74350122)

[Gambar 4. 11 Halaman Detail *Product* 87](#_Toc74350123)

[Gambar 4. 12 Tampilan Detail Produk yang Sudah Diubah 88](#_Toc74350124)

[Gambar 4. 13 Halaman *Admin* Dapat Melihat Detail Produk 88](#_Toc74350125)

[Gambar 4. 14 Halaman Hapus *Product* 89](#_Toc74350126)

[Gambar 4. 15 Halaman CRUD Tabel *Courier* 97](#_Toc74350127)

[Gambar 4. 16 Halaman *Create* pada Tabel *Courier* 97](#_Toc74350128)

[Gambar 4. 17 Halaman CRUD Tabel *Courier* 98](#_Toc74350129)

[Gambar 4. 18 Halaman Detail *Courier* 99](#_Toc74350130)

[Gambar 4. 19 Halaman Detail Data Kurir 99](#_Toc74350131)

[Gambar 4. 20 Tampilan *Admin* Berhasil Menghapus Data pada Tabel *Courier* 100](#_Toc74350132)

[Gambar 4. 21 Tampilan Halaman CRUD Tabel *Category* 106](#_Toc74350133)

[Gambar 4. 22 Tampilan Halaman *Create* *Product* 106](#_Toc74350134)

[Gambar 4. 23 Tampilan Halaman CRUD *Category* 107](#_Toc74350135)

[Gambar 4. 24 Tampilan Halaman Ubah Detail Kategori 108](#_Toc74350136)

[Gambar 4. 25 Tampilan Halaman Detail Kategori yang Berhasil Diubah 108](#_Toc74350137)

[Gambar 4. 26 Tampilan Halaman Hapus Kategori 109](#_Toc74350138)

[Gambar 4. 27 Tampilan Halaman Detail Produk 131](#_Toc74350139)

[Gambar 4. 28 Tampilan Halaman *My Cart* 132](#_Toc74350140)

[Gambar 4. 29 Tampilan Halaman Detail Transaksi 133](#_Toc74350141)

[Gambar 4. 30 Tampilan Halaman *Courier Service* 133](#_Toc74350142)

[Gambar 4. 31 Tampilan Pembatalan Transaksi 139](#_Toc74350143)

[Gambar 4. 32 Tampilan *Upload* Bukti Pembayaran 142](#_Toc74350144)

[Gambar 4. 33 Tampilan *Choose File* 143](#_Toc74350145)

[Gambar 4. 34 Tampilan Bukti Pembayaran yang Telah di *Upload* 143](#_Toc74350146)

[Gambar 4. 35 Tampilan Halaman *My Order* 149](#_Toc74350147)

[Gambar 4. 36 Tampilan Halaman *Transaction* 149](#_Toc74350148)

[Gambar 4. 37 Tampilan Halaman Detail *Transaction* 150](#_Toc74350149)

[Gambar 4. 38 Tampilan *Change Status* 151](#_Toc74350150)

[Gambar 4. 39 Tampilan Halaman Transaksi 151](#_Toc74350151)

[Gambar 4. 40 Tampilan Halaman *Review* 157](#_Toc74350152)

[Gambar 4. 41 Tampilan Halaman *Review* Disimpan 158](#_Toc74350153)

[Gambar 4. 42 Tampilan Halaman *Review* pada *Dashboard Admin* 158](#_Toc74350154)

[Gambar 4. 43 Tampilan Halaman Detail Ulasan 159](#_Toc74350155)

[Gambar 4. 44 Tampilan *Window Leave Response* 159](#_Toc74350156)

[Gambar 4. 45 Tampilan Halaman *List Notifikasi* 168](#_Toc74350157)

[Gambar 4. 46 Tampilan Halaman Laporan Transaksi 175](#_Toc74350158)

# DAFTAR TABEL

[Tabel 1. 1 Sistematika Penulisan Laporan Praktikum Pemrograman Internet 3](#_Toc74478834)

[Tabel 3. 1 Kamus Data *Users* 22](#_Toc74478835)

[Tabel 3. 2 Kamus Data *User\_Notifications* 23](#_Toc74478836)

[Tabel 3. 3 Kamus Data *Admins* 25](#_Toc74478837)

[Tabel 3. 4 Kamus Data *Admin\_Notifications* 26](#_Toc74478838)

[Tabel 3. 5 Kamus Data *Products* 27](#_Toc74478839)

[Tabel 3. 6 Kamus Data *Product\_Reviews* 29](#_Toc74478840)

[Tabel 3. 7 Kamus Data *Product\_Images* 30](#_Toc74478841)

[Tabel 3. 8 Kamus Data *Product\_Categories* 31](#_Toc74478842)

[Tabel 3. 9 Kamus Data *Products\_Category\_Details* 31](#_Toc74478843)

[Tabel 3. 10 Kamus Data *Carts* 32](#_Toc74478844)

[Tabel 3. 11 Kamus Data *Couriers* 33](#_Toc74478845)

[Tabel 3. 12 Kamus Data *Response* 34](#_Toc74478846)

[Tabel 3. 13 Kamus Data *Discounts* 35](#_Toc74478847)

[Tabel 3. 14 Kamus Data *Transactions* 36](#_Toc74478848)

[Tabel 3. 15 Kamus Data *Transaction\_Details* 38](#_Toc74478849)

# DAFTAR KODE PROGRAM

[Kode Program 2. 1 Contoh Penulisan *Tag* HTML 6](#_Toc74490749)

[Kode Program 2. 2 Contoh Penulisan *Tag Head* 7](#_Toc74490750)

[Kode Program 2. 3 Contoh Penulisan *Tag Title* 7](#_Toc74490751)

[Kode Program 2. 4 Contoh Penggunaan *Tag Body* 8](#_Toc74490752)

[Kode Program 2. 5 Contoh Penggunaan *Tag Heading* 9](#_Toc74490753)

[Kode Program 2. 6 Contoh Penggunaan *Tag Horizontal Ruler* 9](#_Toc74490754)

[Kode Program 4. 1 *Source Code* auth.php 43](#_Toc74490755)

[Kode Program 4. 2 *Source Code* admin.php 44](#_Toc74490756)

[Kode Program 4. 3 *Source Code* login.blade.php 46](#_Toc74490757)

[Kode Program 4. 4 *Source Code Login Controller* 47](#_Toc74490758)

[Kode Program 4. 5 *Source Code Login User* 51](#_Toc74490759)

[Kode Program 4. 6 *Source Code View Login* 53](#_Toc74490760)

[Kode Program 4. 7 *Source Code Login Controller* 55](#_Toc74490761)

[Kode Program 4. 8 *Source Code Register User* 58](#_Toc74490762)

[Kode Program 4. 9 *Source Code View Register* 64](#_Toc74490763)

[Kode Program 4. 10 *Source Code Controller Register* 65](#_Toc74490764)

[Kode Program 4. 11 *Source Code Product* 69](#_Toc74490765)

[Kode Program 4. 12 *Source Code View Admin Product* 73](#_Toc74490766)

[Kode Program 4. 13 *Source Product Controller* 77](#_Toc74490767)

[Kode Program 4. 14 *Source Code Courier* 83](#_Toc74490768)

[Kode Program 4. 15 *Source Code* *View Courier* 87](#_Toc74490769)

[Kode Program 4. 16 *Source Code* *Courier Controller* 89](#_Toc74490770)

[Kode Program 4. 17 *Source Code Category* 94](#_Toc74490771)

[Kode Program 4. 18 *Source Code View Admin Category* 96](#_Toc74490772)

[Kode Program 4. 19 *Source Code Category Controller* 98](#_Toc74490773)

[Kode Program 4. 20 *Source Code Cart* 103](#_Toc74490774)

[Kode Program 4. 21 *Source Code View MyCart* 108](#_Toc74490775)

[Kode Program 4. 22 *Source Code View Cart Courier Service* 112](#_Toc74490776)

[Kode Program 4. 23 *Source Code View Cart Payment* 114](#_Toc74490777)

[Kode Program 4. 24 *Source Code View Product Courier Service* 123](#_Toc74490778)

[Kode Program 4. 25 *Source Code Home Controller* 124](#_Toc74490779)

[Kode Program 4. 26 *Source Code Transaction Detail* 127](#_Toc74490780)

[Kode Program 4. 27 *Source Code View Payment Confirmation* 128](#_Toc74490781)

[Kode Program 4. 28 *Source Code* Pembatalan Transaksi 131](#_Toc74490782)

[Kode Program 4. 29 *Source Code Transaction Detail* 133](#_Toc74490783)

[Kode Program 4. 30 *Source Code Upload* Bukti Pembayaran 134](#_Toc74490784)

[Kode Program 4. 31 *Source Code Upload* Bukti Pembayaran 134](#_Toc74490785)

[Kode Program 4. 32 Tampilan Model *Transaction* 138](#_Toc74490786)

[Kode Program 4. 33 Tampilan *My Order List* 139](#_Toc74490787)

[Kode Program 4. 34 Tampilan *Transaction Home* 139](#_Toc74490788)

[Kode Program 4. 35 Tampilan *Transaction Show* 140](#_Toc74490789)

[Kode Program 4. 36 Tampilan *List* Transaksi 141](#_Toc74490790)

[Kode Program 4. 37 Tampilan *Transaction Controller* 141](#_Toc74490791)

[Kode Program 4. 38 *Source Code Product Review* 146](#_Toc74490792)

[Kode Program 4. 39 *Source Code Response* 146](#_Toc74490793)

[Kode Program 4. 40 *Source Code* Detail *Product* 147](#_Toc74490794)

[Kode Program 4. 41 *Source Code Response Home* 148](#_Toc74490795)

[Kode Program 4. 42 *Source Code Response Show* 149](#_Toc74490796)

[Kode Program 4. 43 *Source Code* Tambah *Response* 149](#_Toc74490797)

[Kode Program 4. 44 *Source Code List Review* 149](#_Toc74490798)

[Kode Program 4. 45 *Source Code User Notification* 154](#_Toc74490799)

[Kode Program 4. 46 *Source Code View User Page Notification* 155](#_Toc74490800)

[Kode Program 4. 47 *Source Code Notify User* 155](#_Toc74490801)

[Kode Program 4. 48 *Source Code* Notifikasi Perubahan Status Pesanan 156](#_Toc74490802)

[Kode Program 4. 49 *Source Code List* Notifikasi 156](#_Toc74490803)

[Kode Program 4. 50 *Source Code Admin Notification* 157](#_Toc74490804)

[Kode Program 4. 51 *Source Code View Admin Notification* 158](#_Toc74490805)

[Kode Program 4. 52 *Source Code Notify Controller* 159](#_Toc74490806)

[Kode Program 4. 53 *Source Code* Notifikasi *Proof Of Payment* 159](#_Toc74490807)

[Kode Program 4. 54 *Source Code Transaction* 162](#_Toc74490808)

[Kode Program 4. 55 *Source Code View Home Admin* 166](#_Toc74490809)

[Kode Program 4. 56 *Source Code Admin Controller* 167](#_Toc74490810)

# BAB I PENDAHULUAN

Bab I berisi tentang pendahuluan dari Laporan Akhir Praktikum Pemrograman Internet. Pendahuluan ini akan mengulas secara rinci latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika penulisan dari laporan ini. Rinciannya dapat dilihat sebagai berikut.

## Latar Belakang

Pada era revolusi industri digital 4.0, perkembangan ilmu pengetahuan dan teknologi berjalan dengan pesat. Perkembangan ini juga secara tidak langsung ikut beriringan dengan perkembangan pada bidang-bidang lain dalam kehidupan dan tidak bisa dihindari. Saat ini, teknologi tidak hanya diterapkan pada bidang-bidang tertentu seperti pendidikan saja, namun sudah diterapkan di seluruh aspek kehidupan lainnya.

Internet merupakan jaringan global yang menghubungkan jaringan dengan jaringan lainnya di seluruh dunia. Munculnya internet menjadi salah satu puncak perkembangan ilmu pengetahuan dan teknologi, yang menjadikan aspek-aspek kehidupan lainnya ikut menyesuaikan diri supaya tidak tertinggal. Melalui fitur- fiturnya yang beragam dan keuntungan yang ditawarkan, kini sistem konvensional sudah ditinggalkan dan beralih pada sistem lebih modern yaitu *online*. Pembentukan aplikasi-aplikasi yang menunjang hal tersebut pastinya diperlukan agar mampu menerapkan hal tersebut dalam kehidupan sehari-hari.

Penerapan aplikasi berbasis internet bisa dilakukan dengan menggunakan pemrograman internet. Pemrograman internet atau pemrograman berbasis internet adalah suatu jenis aplikasi yang menerapkan arsitektur sistem terdistribusi. Bahasa pemrograman yang populer digunakan untuk membangun suatu aplikasi internet berbasis web yang sifatnya dinamis yaitu PHP. Selain itu, terdapat juga beberapa *framework* yang sering digunakan oleh para *developer* web dengan PHP seperti Laravel yang merupakan *framework* PHP pertama yang pernah dirilis dengan pengguna paling banyak.

## Rumusan Masalah

Berdasarkan latar belakang yang telah disampaikan pada poin 1.1, dapat disimpulkan beberapa rumusan masalah yang akan dipecahkan pada Laporan Praktikum Pemrograman Internet ini. Berikut adalah beberapa rumusan masalah pada Laporan Praktikum Pemrograman Internet ini.

1. Bagaimana cara penerapan *multi-authentication* pada *framework* Laravel.
2. Bagaimana cara penerapan CRUD (*Create, Read, Update, Delete*) pada *framework* Laravel.
3. Bagaimana cara mengimplementasikan proses transaksi pada rancangan sistem *web* menggunakan *framework* Laravel.
4. Bagaimana cara mengimplementasikan sistem notifikasi dan *report* pada rancangan sisten *web* menggunakan *framework* Laravel.

## 1.3 Tujuan

Berdasarkan rumusan-rumusan masalah yang telah disebutkan pada poin 1.2, dapat diambil kesimpulan beberapa tujuan dari pembuatan Laporan Praktikum Pemrograman Internet ini. Berikut merupakan beberapa tujuan dari pembuatan laporan ini.

1. Untuk mengetahui cara penerapan *multi-authentication* pada *framework* Laravel.
2. Untuk mengetahui cara penerapan CRUD (*Create, Read, Update,* dan *Delete*) pada *framework* Laravel.
3. Untuk mengetahuicara mengimplementasikan proses transaksi pada rancangan sistem *web* menggunakan *framework* Laravel.
4. Untuk mengetahui cara mengimplementasikan sistem notifikasi pada rancangan sistem *web* menggunakan *framework* Laravel.

## 1.4 Manfaat

Beberapa manfaat yang dapat diambil berdasarkan tujuan pembuatan Laporan Praktikum Pemrograman Internet ini antara lain mengetahui dan memahami dasar-dasar penggunaan *framework* Laravel untuk membangun sebuah aplikasi berbasis *web*, penerapan *multi-authentication*, CRUD, dan implementasi transaksi serta sistem notifikasi pada sistem berbasis *web* yang telah dibangun menggunakan *framework* Laravel.

## 1.5 Batasan Masalah

Penyusunan Laporan Praktikum Pemrograman Internet ini dibatasi oleh beberapa masalah seperti penerapan *multi-authentication* dan CRUD serta penerapan transaksi menggunakan sistem *web* dan sistem notifikasi pada *web* yang telah dibangun menggunakan *framework* Laravel.

## 1.6 Sistematika Penulisan

Sistematika penulisan digunakan sebagai acuan dalam penyusunan Laporan Praktikum Pemrograman Internet. Sistematika penulisan yang digunakan dalam menyusun laporan ini dapat dilihat pada tabel berikut.

Tabel 1. 1 Sistematika Penulisan Laporan Praktikum Pemrograman Internet

|  |  |
| --- | --- |
| **Bab** | **Keterangan** |
| Bab I Pendahuluan | Membahas latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan laporan. |
| Bab II Tinjauan Pustaka | Membahas tinjauan pustaka secara umum yang digunakan sebagai referensi dalam pengerjaan modul-modul dan laporan Praktikum Pemrograman Internet |
| Bab III Metode Penelitian | Membahas tempat dan waktu pembuatan penelitian, gambaran umum sistem yang akan dibangun, serta rancangan database yang diterapkan pada sistem yang telah dibangun. |
| Bab IV Pembahasan dan Hasil Analisa | Membahas isi-isi pokok modul 1 hingga modul 5 pada praktikum Pemrograman Internet. |
| Bab V Penutup | Menyimpulkan hasil analisa yang dilakukan dengan mengaitkannya dengan rumusan masalah yang telah dijabarkan pada pendahuluan serta tinjauan pustaka yang telah dilakukan sebelumnya serta saran-saran untuk memaksimalkan praktikum lain ke depannya. |

Tabel 1. 1 merupakan tabel yang berisi sistematika penulisan Laporan Praktikum Pemrograman Internet yang terdiri dari lima bab yaitu Pendahuluan, Tinjauan Pustaka, Metode Penelitian, Pembahasan dan Hasil Analisa, serta Penutup.

# 

# BAB II TINJAUAN PUSTAKA

Bab II berisi tinjauan pustaka sebagai sumber literasi untuk penyusunan Laporan Akhir Praktikum Pemrograman Internet. Pembahasan secara rinci tentang tinjauan pustaka ini adalah tentang apa itu pemrograman dan pemrograman internet, *framework* Laravel, basis data, *multiauthentication,* dan proses CRUD atau *Create, Read, Update,* dan *Delete* khususnya pada Laravel.

## 2.1 Pemrograman Internet

Internet yang merupakan singkatan dari *Interconnected Network* adalah sebuah sistem komunikasi global yang menghubungkan komputer-komputer dan jaringan-jaringan komputer di seluruh dunia. Setiap komputer atau jaringan tersebut terhubung secara langsung maupun tidak langsung ke dalam beberap jalur utama yang disebut dengan Internet *Backbone*. Masing-masing komputer yang terhubung dapat dibedakan satu dengan lainnya menggunakan sebuah *unique name* yang biasa disebut IP *Address*. (Waslaluddin, 2010).

Pemrograman merupakan suatu proses iteratif penulisan dan penyuntingan kode sumber sehingga membentuk sebuah program. Penyuntingan kode sumber meliputi proses pengetesan, analisis, pembetulan kesalahan, pengoptimasian algoritma, normalisasi kode, dan pengkoordinasian antara satu pemrogram dengan yang lainnya. Terdapat beberapa jenis pemrograman berdasarkan basisnya, salah satunya yaitu Pemrograman Internet. Pemrograman internet merupakan jenis pemrograman aplikasi yang berbasis internet.

## 2.2 HTML

HTML merupakan singkatan dari *hypertext markup language* adalah salah satu format yang digunakan dalam pembuatan dokumen (*web page*) dan aplikasi yang berjalan di halaman web. HTML disebut sebagai *markup language* karena dalam teks HTML mengandung beberapa *tag* tertentu yang digunakan untuk menentukan tampilan suatu teks dan tingat kepentingan teks tersebut dalam suatu dokumen. Dokumen HTML merupakan dokumen yang disajikan pada *web browser*. Ekstensi dari file HTML umumnya yaitu .html atau .htm. HTML bersifat *multi-platform* atau dapat dijalankan pada sistem operasi manapun. Penulisan dokumen HTML dapat dilakukan menggunakan *text editor* seperti Notepad, Visual Studio Code, dan lain sebagainya, maupun menggunakan *web browser* seperti Mozilla Firefox, Google Chrome, dan lainnya. Terdapat beberapa ciri-ciri dari dokumen HTML di antaranya sebagai berikut.

1. Terdapat tag HTML yang diawali dengan tag pembuka dan diakhiri tag penutup.
2. Terdapat elemen HTML yang terdiri dari dua bagian yaitu *head* untuk memberi informasi tentang dokumen dan *body* untuk menentukan bagaimana isi suatu dokumen yang akan ditampilkan pada *web browser*.

Seperti yang telah dijelaskan pada poin pertama ciri-ciri dari dokumen HTML yaitu terdapat *tag* HTML, berikut ini akan dijelaskan beberapa *tag* dasar dalam dokumen HTML.

### 2.2.1 Tag Dasar HTML

*Tag* adalah sebuah kode yang digunakan untuk melakukan *mark up* teks ASCII menjadi *file* HTML. Setiap *tag* diapit dengan tanda kurung runcing (< ... >). Ada *tag* pembuka yaitu <HTML> dan ada *tag* penutup </HTML> yang ditandai dengan tanda *slash* (/). *Tag* tersebut memberikan kaidah bahwa yang ditulis di antara kedua *tag* tersebut adalah isi dari dokumen HTML. Berikut ini merupakan beberapa *tag* dasar HTML.

#### 2.2.1.1 HTML

HTML merupakan *tag* dasar yang mendefinisikan bahwa suatu dokumen merupakan dokumen HTML. *Tag* ini merupakan *tag* pertama dalam dokumen HTML. Penulisan *tag* HTML dapat dilihat pada kode program berikut.

|  |
| --- |
| <html>  .....  </html> |

Kode Program 2. 1 Contoh Penulisan *Tag* HTML

Kode Program 2. 1 menunjukkan cara penulisan *tag* HTML yang akan mengawali sebuah dokumen HTML dan mengakhiri dokumen tersebut. *Tag* ini merupakan *tag* yang mendefinisikan suatu dokumen sebagai dokumen HTML.

#### 2.2.1.2 Head

Jenis *tag* HTML lainnya yaitu *tag head* yang merupakan *tag* setelah *tag* HTML yang berisi tentang keterangan tentang dokumen yang akan ditampilkan. Penulisan *tag head* dapat dilihat pada kode program berikut.

|  |
| --- |
| <html>  <head>  .....  </head>  </html> |

Kode Program 2. 2 Contoh Penulisan *Tag Head*

Kode Program 2. 2 menunjukkan cara penulisan *tag head* dalam dokumen HTML yang berisi tentang keterangan dokumen yang akan ditampilkan pada sebuah *web page*.

#### 2.2.1.3 Title

Jenis *tag* HTML lainnya yaitu *tag title* yang merupakan *tag* yang ada pada sebuah *head* yang digunakan untuk menuliskan judul dari dokumen HTML. Judul tersebut akan muncul pada *caption* halaman *browser* jika halaman tersbeut diakses. Penulisan *tag title* dapat dilihat pada kode program berikut.

|  |
| --- |
| <html>  <head>  <title>WEB LARAVEL KELOMPOK 20</title>  </head>  </html> |

Kode Program 2. 3 Contoh Penulisan *Tag Title*

Kode Program 2. 3 menunjukkan cara penulisan *tag title* dalam dokumenHTML yang digunakan untuk menuliskan judul dari dokumen HTML yang akan diakses.

#### 2.2.1.4 Body

Jenis *tag* umum lainnya pada dokumen HTML yaitu *tag body* yang merupakan bagian utama dari sebuah halaman *web*. Bagian dalam *tag body* akan berisi semua isi dokumen yang ingin ditampilkan di dalam sebuah *browser* ketika diakses. Penulisan *tag body* dapat dilihat pada kode program berikut.

|  |
| --- |
| <html>  <head>  <title>WEB LARAVEL KELOMPOK 20</title>  </head>  <body>  Welcome to Laravel Website Of Kelompok 20 Praktikum Pemrograman Internet  </body>  </html> |

Kode Program 2. 4 Contoh Penggunaan *Tag Body*

Kode Program 2. 4 merupakan contoh penggunaan *tag body* dalam dokumen HTML dimana di dalamnya berisikan semua dokumen yang akan ditampilkan pada halaman *web*.

#### 2.2.1.5 Heading

Jenis tag HTML lainnya yaitu *tag heading* yang digunakan untuk membuat format *heading* (judul atau sub judul) dari halaman *web*. Terdapat 6 jenis heading yang ada di dalam HTML. Contoh penulisan *tag heading* dapat dilihat pada kode program berikut.

|  |
| --- |
| <html>  <head>  <title>WEB LARAVEL KELOMPOK 20</title>  </head>  <body>  <h1> Heading 1 </h1>  <h2> Heading 2 </h2>  <h3> Heading 3 </h3>  <h4> Heading 4 </h4>  <h5> Heading 5 </h5>  <h6> Heading 6 </h6>  </body>  </html> |

Kode Program 2. 5 Contoh Penggunaan *Tag Heading*

Kode Program 2. 5 merupakan contoh cara penulisan *tag heading* dalam dokumen HTML yang berguna untuk membuat judul atau sub judul dalam dokumen HTML yang akan ditampilkan di halaman *web*.

#### 2.2.1.6 Horizontal Ruler

*Tag horizontal ruler* atau <hr> berfungsi untuk menampilkan garis horizontal tiga dimensi di dalam sebuah halaman *web*. *Tag horizontal ruler* ini tidak memerlukan elemen penutup </hr>. Terdapat sejumlah atribut dalam horizontal ruler seperti *align* (dengan *default center*), *size*, dan *width*. Berikut merupakan contoh penulisan *tag horizontal ruller*.

|  |
| --- |
| <html>  <head>  <title>WEB LARAVEL KELOMPOK 20</title>  </head>  <body>  <h1> Heading 1 </h1>  <hr width=”50%”>  </body>  </html> |

Kode Program 2. 6 Contoh Penggunaan *Tag Horizontal Ruler*

Kode Program 2. 6 merupakan salah satu contoh penggunaan *tag horizontal ruler* pada dokumen HTML yang berguna untuk mengatur penulisan judul aau sub judul dalam dokumen HTML di sebuah halaman *web*.

#### 2.2.1.7 Paragraph

*Tag paragraph* memiliki fungsi seperti pengaturan antar paragraf dalam sebuah *web*. Pada elemen paragraf, terdapat beberapa atriut seperti *align* untuk mengatur perataan paragraf.

#### 2.2.1.8 List

*List item* merupakan sebuah perintah yang digunakan dalam HTML untuk membuat daftar atau mengelompokkan kata. Terdapat tiga jenis *list item* dalam dokumen HTML, yaitu *ordered list*, *unordered list*, dan *break*. *Ordered list* digunakan untuk membuat daftar dimana setiap bagiannya memiliki nomor secara urut. *Unordered list* digunakan untuk membuat daftar tanpa nomor yang urut. Sementara *break* merupakan *tag* yang berfungsi untuk memberikan sebuah garis baru pada sebuah paragraf dalam halaman *web*.

## 2.3 PHP

PHP merupakan singkatan dari *hypertext prepocessor*. PHP adalah bahasa *script* yang dijalankan pada sisi *server* atau *server side scripting*. Kelebihan PHP sebagai bahasa *script* adalah dapat dijalankan pada berbagai platform baik itu Windows atau Linux, kompatibel terhadap hampir semua jenis *server*, mudah dipelajari, dan berjalan dengan efisien pada sisi *server*. Untuk dapat menggunakan PHP, pengguna harus menginstal *web server* Apache terlebih dahulu pada *server* yang digunakan serta melakukan instalasi PHP dan MySQL. PHP merupakan sebuah *software open source* yang bebas untuk diunduh dan digunakan.

PHP bisa di-*embed* ke dalam HTML atau bisa dikombinasikan dengan berbagai sistem *template web*, *content* *management system*, dan *framework* sebuah *web*. *Framework* sendiri dalam bahasa Indonesia disebuat sebagai kerangka kerja yaitu *software* yang memudahkan para *programmer* untuk membuat aplikasi atau *web* yang disinya adalah berbagai fungsi, *plugin*, dan konsep yang akan membentuk sistem tertentu. Dengan menggunakan *framework*, sebuah aplikasi akan tersusun dan terstruktur dengan rapi.

Jika sebuah *project* menggunakan *framework* PHP, maka digunakan metode pengembangan berbasis MVC. MVC sendiri merupakan sebuah metode untuk memisahkan bagian-bagian dari suatu *web* aplikasi. MVC merupakan singkatan dari *model*, *view*, dan *controller*. *Model* merupakan model yang mewakili sebuah struktur data, biasanya model berisi fungsi-fungsi yang membantu *programmer* dalam pengelolaan *database* seperti proses CRUD. *View* merupakan bagian yang digunakan untuk mengatur bagaimana tampilan yang dikeluarkan dari aplikasi atau *web* kepada *user*. Sementara, *controller* merupakan bagian yang menjembatani *model* dan *view*. *Controller* biasanya berisi *script* PHP yang berfungsi memproses suatu data dan mengirimkannya ke halaman *web*.

*Franework* sendiri memiliki beberapa keunggulan seperti ringan dan cepat, hal ini karena *framework* hanya melakukan pemanggilan pustaka atau kelas yang dibutuhkan sehingga meminimalkan sumber daya yang diperlukan, hal tersebut membuat *load* sebuah halaman akan menjadi lebih ringan dan cepat. Keunggulan lainnya yaitu menggunakan metode MVC sehingga masalah-masalah seperti bagian tampilan, logika, dan *query* *database* terpecah dan lebih mudah melakukan perombakan salah satu bagian. Pada bahasa pemrograman PHP sendiri terdapat sejumlah *framework* yang dapat diterapkan, adapun *framework* yang sering digunakan yaitu *framework* Laravel yang akan dijelaskan sebagai berikut.

### 2.3.1 Laravel

*Framework Laravel* merupakan PHP *framework* yang pertama kali dirilis pada tahun 2011. Berdasarkan survei oleh Sitepoint, Laravel merupakan *framework* yang paling banyak digunakan oleh *developer*. Laravel sendiri memiliki beberapa fitur yang dapat membantu *developer* untuk membuat aplikasi dengan lebih cepat. Laravel memiliki mesin *templating* ringan yang diberi nama *blade*. Tidak hanya itu, Laravel juga memiliki sintaks yang bisa membantu pihak pengembang untuk melakukan *task* seperti autentifikasi, *sessions*, *queueing*, *caching*, dan *routing*.

#### 2.3.1.1 Kelebihan Laravel

*Framework Laravel* dibangun dengan menggunakan konsep MVC dan dilengkapi dengan sebuah *command line tool* yang diberi nama “artisan” yang dapat digunakan untuk *packaging bundle* dan *instalasi bundle* melalui *Command Prompt*. *Framework* ini menekankan kesederhanaan dan fleksibilitas pada desainnya. Berikut merupakan beberapa kelebihan dari Laravel.

1. Ekspresif

Laravel adalah *framework* PHP yang ekspresif, artinya ketika melakukan suatu sintaks, *programmer* diharapkan dapat langsung tahu kegunaan dari sintaks tersebut meskipun belum pernah mempelajarinya. Laravel menggunakan pendekatan yang berbeda dengan membuat parameter yang sifatnya *flagging* dan memilih untuk membuat dua fungsi yang berbeda.

1. Simple

Salah satu alasan yang membuat Laravel sederhana atau *simple* yaitu dengan adanya *Eloquent ORM*. Ketika *programmer* ingin mengambil semua data yang ada di dalam tabel *user*, maka yang diperlukan hanya membuat sebuah *class model* bernama *user*, kemudian tinggal memasukkan data dari tabel *user* ke dalamnya. Laravel juga disebut sederhana dalam masalah routing-nya.

1. Accessible

*Framework* Laravel mudah digunakan baik bagi *programmer* pemula, sebab *framework* ini memiliki dokumentasi yang sangat lengkap. *Developer* dari *framework* ini berkomitmen untuk selalu menyertakan dokumentasi yang lengkap setiap kali melakukan rilis versi terbarunya.

#### 2.3.1.2 Fitur-fitur Laravel

*Framework* Laravel memiliki beberapa fitur yang memudahkan *developer* atau *programmer* untuk membuat aplikasi atau web. Fitur-fitur tersebut antara lain sebagai berikut.

1. *Bundles*, sebuah fitur dengan sistem pengemasan modular.
2. *Eloquent ORM*, merupakan penerapan PHP lanjutan dari pola *active record* yang menyediakan metode internal untuk mengatasi kendala hubungan antar objek *database*.
3. *Application Logic*, merupakan bagian dari aplikasi yang dikembangkan, baik menggunakan *controller* maupun bagian dari deklarasi *route*.
4. *Reverse Routing*, berguna untuk mendefinisikan hubungan antara *link* dengan *route*, sehingga jika suatu saat ada perubahan pada *route*, secara otomatis akan tersambung dengan *link* yang relevan.
5. *Restful Controller*, memberikan sebuah pilihan untuk memisahkan logika dalam melayani HTTP GET dan permintaan POST.
6. *Class Auto Loading*, menyediakan otomatis loading untuk *class* PHP tanpa membutuhkan pemeriksaan manual terhadap jalur masuknya. Fitur ini dapat mencegah adanya *loading* yang tidak diperlukan.
7. *View Composer*, merupakan kode unit logika yang dapat dijalankan ketika sebuah view di-*load*.
8. *IoC Container*, memungkinkan untuk objek baru yang dihasilkan dengan mengikuti konsep *control* pembalik, dengan pilihan contoh dan referensi dari objek baru sebagai *Singletons*.
9. *Migrations*, menyediakan versi sistem kontrol untuk skema *database*, sehingga memungkinkan untuk menghubungkan perubahan.
10. *Unit Testing*, memiliki peran penting dalam *framework* Laravel, yaitu mempunyai banyak tes untuk melakukan deteksi dan mencegah regresi.
11. *Automatic Pagination*, menyederhanakan tugas dari penerapan halaman, menggantikan penerapan yang manual dengan metode otomatis yang terintegrasi ke Laravel.

#### 2.3.1.3 Konfigurasi Sistem

Untuk *developer* pemula, terdapat sejumlah konfigurasi sistem yang dilakukan dalam pembuatan proyek aplikasi *web* ataupun *website* menggunakan *framework* Laravel. Konfigurasi-konfigurasi tersebut antara lain sebagai berikut.

1. Konfigurasi *database*, *cache*, *mail*, dan *setting* lainnya sesuai dengan kebutuhan.
2. Menciptakan *routing* aplikasi.
3. Membuat model dan struktur *database* untuk data.
4. Membuat *controller* dan mengintegrasikannya dengan *routing*.
5. Membuat *view template*.
6. Pengujian aplikasi.
7. Penyempurnaan kode aplikasi.

#### 2.3.1.4 Kebutuhan Sistem

Beberapa kebutuhan sistem yang perlu diinstal pada *server* oleh *developer* dalam pembuatan proyek aplikasi *web* maupun *website* dengan menggunakan *framework* Laravel adalah sebagai berikut.

1. Text Editor
2. Web Server
3. OpenSLL
4. Composer Laravel

## 2.4 CSS

CSS adalah bahasa *cascading style sheet* yang biasanya digunakan untuk mengatur tampilan elemen yang tertulis dalam bahasa *markup* seperti HTML. CSS berfungsi untuk memisahkan konten dari tampilan visual di situs. CSS dibuat dan dikembangkan oleh W3C atau *world wide web consortium* pada tahun 1996. CSS menggunakan bahasa inggris sederhana berbasis sintaks yang dilengkapi dengan sekumpulan *rule* yang mengaturnya.

Struktur dari sintaks CSS cukup sederhana, dimana struktur ini memiliki *selector* dan *declaration block*. *Selector* mengarah pada elemen HTML yang ingin diubah tampilannya, sementara *declaration block* memuat satu atau lebih deklarasi yang dipisahkan dengan tanda titik koma (;).

## 2.5 JavaScript

JavaScript digunakan untuk menjadikan *website* lebih interaktif. *Script* ini dijalankan pada *browser user*, dan biasanya masuk ke library pihak ketiga untuk menyediakan fungsionalitas tingkat lanjut tanpa mengharuskan *developer* melakukan *coding* dari awal. JavaScript dibuat dan didesain selama sepuluh hari oleh Brandan Eich pada bulan September 1995. Kurun waktu dua puluh tahun berikutnya, JavaScript beralih dari bahasa pemrograman yang serba terbatas dan primitif menjadi salah satu *tool* terpenting bagi *web developer*.

Terdapat sejumlah kelebihan dari JavaScript seperti *developer* tidak membutuhkan *compiler* karena *browser* mampu menginterpretasikannya dengan HTML, lebih mudah dipelajari dibandingkan dengan bahasa pemrograman lainnya, kesalahan lebih mudah dicari dan ditangani, serta sifatnya *multi-platform*. Selain itu terdapat kekurangan dari JavaScript seperti berisiko tinggi terhadap eksploitasi, dapat dimanfaatkan untuk mengaktifkan kode berbahaya di komputer pengguna, dan tidak selalu didukung oleh berbagai *browser* dan perangkat.

## 2.6 JQuery

JQuery adalah library JavaScript dimana bahasa pemrograman ini dibuat oleh John Resig tahun 2006, untuk memudahkan para *developer* dalam menggunakan dan menerapkan JavaScript di *website*. Fungsi dari JQuery adalah mengompres berbagai baris kode dalam satu buah fungsi sehingga *developer* tidak perlu menulis kembali semua baris kode hanya untuk menyelesaikan sebuah *task*. JQuery memiliki beberapa fitur unggulan seperti manipulasi HTML/DOM, manipulasi CSS, HTML *event method*, efek dan animasi, AJAX, CSS3 *compliant*, dan *cross-browser*.

JQuery mendapat cukup banyak dukungan dalam dunia *open source* karena mempunyai banyak kelebihan seperti dapat berinteraksi baik dengan berbagai macam tipe bahasa pemrograman lain, mendukung berbagai macam *plugin*, dan membuat sebuah animasi dasar dengan sangat mudah. Meskipun mempunyai banyak keunggulan, JQuery juga mempunyai beberapa kekurangan seperti perangkat lunak *open source* terkadang mempunyai beberapa masalah, beberapa versi JQuery sudah tidak kompatibel (perlu diperbaharui beberapa kali), dan fitur *lightweight* JQuery memungkinkan timbulnya sebuah permasalahan.

## 2.7 MySQL

MySQL merupakan salah satu *Relational Database Management System* (RDBMS) dengan *client-server model* dimana pengembangannya berada di bawah Oracle yang dirilis pada tahun 1995 yang menggunakan perintah dasar SQL (*Structured Query Language*) sebagai bahasa penghubung antara perangkat lunak aplikasi dengan *database*. MySQL merupakan *database server* gratis dengan lisensi *General Public License* (GPL) sehingga dapat digunakan untuk keperluan pribadi ataupun komersil tanpa harus membayar lisensi yang ada. MySQL memiliki beberapa keistimewaan seperi berikut ini.

1. MySQL dapat berjalan dengan stabil pada berbagai sistem operasi seperti Windows, Linux, Mac OS X Server, FreeBSD, Solaris, Amiga dan masih banyak lagi.
2. MySQL didistribusikan sebagai perangkat lunak *opern source* yang berada dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. MySQL memiliki kemampuan untuk menangani *query* sederhana dengan waktu yang singkat.
5. MySQL memiliki jenis tipe data yang beragam seperti *integer*, *float*, *double*, *char*, *text*, *date*, *timestamp*, dan lain-lain.
6. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam *query*.
7. MySQL memiliki beberapa lapisan keamanan seperti *level subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. MySQL mampu menangani *database* dalam skala besar, dengan jumlah *records* lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris, selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix Soket (UNIX), atau Named Pipes.
10. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa, namun untuk saat ini bahasa Indonesia belum termasuk di dalamnya.
11. MySQL dilengkapi dengan berbagai *tools* yang dapat digunakan untuk administrasi *database*, dan pada setiap peralatan yang ada disertakan petunjuk *online*.
12. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani *alter* tabel, dibandingkan *database* lainnya seperti PostgreSQL ataupun Oracle.

## 2.8 Git dan Github

Git merupakan sebuah *tools* yang digunakan dalam *project* pengembangan *software*. Git adalah salah satu *version control system* (VCS) pada proyek perangkat lunak yang diciptakan oleh Linus Torvald. VCS sendiri memiliki tugas untuk mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang ataupun individual.

Git dikenal juga dengan sebutan VCS terdistribusi atau *distributed revision control* yang artinya penyimpanan *database* Git tidak hanya berada di satu tempat saja. Semua orang yang terlibat dalam suatu *project* Git dapat menyimpan *database* dari *project* Git tersebut sehingga memudahkan dalam hal mengelola proyek baik *online* maupun *offline*. Pada Git terdapat *merge* yaitu sebutan untuk aktivitas penggabungan kode.

Sementara itu Github merupakan manajemen proyek dan sistem VCS sekaligus *platform* jejaring sosial yang dirancang khusus bagi para pengembang perangkat lunak. Sistem Hub pada Github berfungsi untuk mengubah baris perintah seperti Git, menjadi jaringan media sosial terbesar bagi kaum *developer*. Adapun beberapa istilah yang perlu diketahui dalam Github yaitu sebagai berikut.

### 2.8.1 Repositori

*Repositori* atau repo merupakan sebuah direktori yang dapat dibuat oleh pengguna Github untuk menyimpan suatu *file project*. Pengguna bisa menyimpan *file* apapun yang berkaitan dengan proyek yang sedang dikerjakan seperti kode program, gambar, maupun audio. Repo bertempat pada penyimpanan atau *storage* dari Github maupun tersimpan di lokal komputer pengguna.

### 2.8.2 Branch

*Branch* merupakan salinan dari repositor dimana *branch* digunakan ketika *user* hendak melakukan suatu pengembangan secara terpisah. Pekerjaan atau *task* yang dilakukan di *branch* tidak akan memengaruhi repositori pusat maupun *branch* lainnya. Jika *development* telah selesai dan pengguna ingin menggabungkan *branch* dengan repo pusat dapat dilakukan dengan menggunakan *pull request*.

### 2.8.3 Pull Request

*Pull request* adalah sebuah istilah ketika pengguna ingin menginformasikan bahwa ia ingin mengirimkan perubahan yang telah dilakukan pada *branch* ke dalam *master* repo. *Collaborator* akan menerima atau menolak permintaan tersebut, jika diterima maka data dari *branch* akan disinkronisasikan dengan repo pusat.

### 2.8.4 Forking Repositori

*Forking repositori* artinya pengguna membuat sebuah *project* baru berdasarkan *repositori* yang telah ada atau sederhananya pengguna menyalin *repositori* yang telah ada kemudian melakukan perubahan yang diperlukan dan menyimpannya dalam *repositori* baru dan menjadikannya proyek sendiri.

# BAB III METODE PENELITIAN

Bab III berisikan metodologi penelitian dari Laporan Akhir Praktikum Pemrograman Internet. Bab ini akan mengulas secara rinci metode penelitian yang digunakan dalam menyusun laporan praktikum ini antara lain sebagai berikut.

## 3.1 Tempat dan Waktu Penelitian

Penelitian yang dilakukan untuk menyelesaikan Laporan Praktikum Pemrograman Internet ini mengambil tempat di gedung Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana Bukit Jimbaran dengan waktu menyesuaikan jam kuliah (di luar jam kuliah).

## 3.2 Gambaran Umum Sistem

Gambaran umum sistem bertujuan untuk menunjukkan alur kerja dari sistem secara umum yang akan dibuat pada Praktikum Pemrograman Internet. Sistem yang akan dibuat yaitu sebuah sistem *e-commerce* dimana *user* yang terdaftar dibagi menjadi dua kategori dengan akses yang berbeda yaitu *user* biasa serta *admin*. Dari sisi *user* biasa, dapat melakukan proses transaksi pembelian barang dan mendapatkan notifikasi dari sistem *e-commerce*. Sementara, dari sisi *admin* dapat melakukan pendaftaran barang-barang, kurir, dan hal lain yang berhubungan dengan *e-commerce* serta dapat menerima data transaksi *user* untuk selanjutnya melakukan verifikasi pemesanan.

|  |
| --- |
|  |

Gambar 3. 1 Gambaran Umum Sistem

Gambar 3. 1 merupakan diagram yang menampilkan gambaran umum sistem, dimana terdapat dua peran yaitu *admin* dan *user* yang terhubung dengan internet. *Admin* mengirimkan data *admin*, verifikasi transaksi, dan respon ulasan. Sementara *user* mengirimkan data *user*, data pemesanan, butki pembayaran, dan ulasan produk ke *server website e-commerce*, lalu disimpan ke dalam *database*. Sementara itu *server* mengirimkan sejumlah notifikasi kepada *user* dan *admin*.

## 3.3 Perancangan Database

Laporan Praktikum Pemrograman Internet ini disusun berdasarkan penelitian atas lima buah modul yang pada intinya merupakan implementasi dari suatu rancangan aplikasi penjualan atau *e-commerce* berbasis *web* yang wajib mengimplementasikan rancangan *database* yang akan ditampilkan dalam bentuk PDM dan dilengkapi dengan kamus data berikut ini.

### 3.3.1 PDM

*Physical Data Model* atau PDM merupakan gambaran detail dari sebuah basis data, dimana penggambaran sebuah PDM dapat memperlihatkan struktur dari *database* yang sesungguhnya. Sementara itu, *database* atau basis data sendiri berarti kumpulan data yang saling berhubungan satu sama lainnya. Penggambaran rancangan *database* yang digunakan untuk menerapkan aplikasi *e-commerce* berbasis *web* dapat pada gambar berikut ini.

|  |
| --- |
|  |

Gambar 3. 2 Rancangan PDM *E-Commerce* Berbasis Web

Gambar 3. 2 merupakan rancangan PDM dari basis data aplikasi penjualan atau *e-commerce* berbasis web, dimana dalam basis data tersebut terdapat lima belas entitas yang masing-masing saling berelasi satu sama lain. Untuk dapat memahami atribut dari masing-masing entitas tersebut, akan diberikan kamus data dari rancangan *database* aplikasi penjualan berbasis web tersebut.

### 3.3.2 Kamus Data

*Data dictitonary* atau dalam bahasa Indonesia disebut kamus adalah sebuah penjelasan tertulis tentang suatu data yang berada dalam sebuah *database*. Kamus data memuat daftar elemen data yang terorganisir dengan definisi yang tetap dan sesuai dengan sistem, sehingga pengguna dan analis sistem mempunyai pengertian yang sama tentang *input*, *output*, atau komponen lain dalam *datastore*. Berikut ini merupakan kamus data untuk masing-masing entitas dalam rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web.

Tabel 3. 1 Kamus Data *Users*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari masing-masing *user* yang terdaftar di aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *users*. Tipe data dari *field* ini adalah INT (*integer*). |
| 2 | name | VARCHAR[100] | *Field* ini berisi data nama masing-masing *user* yang terdaftar di aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 3 | email | VARCHAR[100] | *Field* ini berisi data alamat surel dari masing-masing *user* yang terdaftar dalam aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 4 | profil\_image | VARCHAR[100] | *Field* ini berisi data URL dari foto profil yang digunakan *user* pada profil yang berhasil didaftarkan. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 5 | status | VARCHAR[100] | *Field* ini berisi status dari akun yang didaftarkan pada sistem *e-commerce* ini dengan *default status* “aktif”. |
| 6 | email\_verified\_at | TIMESTAMP | *Field* ini berisi tanggal dan waktu saat *e-mail* pengguna yang mendaftar telah diverifikasi dengan tipe data TIMESTAMP. |
| 7 | password | VARCHAR[100] | *Field* ini berisi kata sandi ATAU *password* dari profil pengguna yang berhasil didaftarkan pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 8 | remember\_token | VARCHAR[100] | *Field* ini berisi token yang digunakan jika pengguna melupakan *password* yang telah dibuat sebelumnya. *Field* ini bertipe data VARCHAR dengan panjang data 100 karakter. |
| 9 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu akun profil pengguna telah berhasil didaftarkan pada aplikasi *e-commerce*. Tipe data dari field ini adalah TIMESTAMP. |
| 10 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika terjadi pembaharuan profil akun pengguna pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah TIMESTAMP. |

Tabel 3. 1 merupakan kamus data entitas *user* yang memuat atribut atau *field* yang ada pada entitas *users* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *users*, *field* id dipakai sebagai *primary key*.

Tabel 3. 2 Kamus Data *User\_Notifications*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari notifikasi  yang diterima pengguna pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *users\_notifications*. Tipe data dari *field* ini adalah INT. |
| 2 | type | VARCHAR[100] | *Field* ini berisi tipe notifikasi yang diterima *user* pada aplikasi *e- commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 3 | notifiable\_type | VARCHAR[100] | *Field* ini berisi tipe penerima notifikasi pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 4 | notifiable\_id | INT | *Field* ini berisi id pengguna yang menerima notifikasi pada aplikasi *e- commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *users*. |
| 5 | data | TEXT | *Field* ini berisi teks yang akan ditampilkan pada notifikasi yang diterima. Tipe data yang digunakan pada *field* ini adalah TEXT. |
| 6 | read\_at | TIMESTAMP | *Field* ini berisi keterangan waktu saat notifikasi telah dibaca oleh penerima (pengguna) dengan tipe data TIMESTAMP. |
| 7 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu notifikasi penerima dibuat dengan tipe data TIMESTAMP. |
| 8 | Updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika notifikasi pengguna telah diperbaharui dengan tipe data TIMESTAMP. |

Tabel 3. 2 merupakan kamus data entitas *user\_notifications* yang memuat atribut atau *field* yang ada pada entitas *user\_notifications* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *user\_notifications*, *field* id dipakai sebagai *primary key*.

Tabel 3. 3 Kamus Data *Admins*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari *admin* pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *admins*. Tipe data dari *field* ini adalah INT. |
| 2 | username | VARCHAR[100] | *Field* ini berisi data *username* dari *admin* pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 3 | password | VARCHAR[100] | *Field* ini berisi data kata sandi profil *admin* pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 4 | name | VARCHAR[100] | *Field* ini berisi data nama dari *admin* pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 5 | profile\_image | VARCHAR[100] | *Field* ini berisi data URL dari foto profil akun *admin* pada aplikasi *e- commerce* dengan menggunakan tipe data VARCHAR dan panjang data 100 karakter. |
| 6 | phone | VARCHAR[12] | *Field* ini berisi data nomor telepon *admin* pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 12 karakter. |
| 7 | remember\_token | VARCHAR[100] | *Field* ini berisi token yang digunakan jika *admin* melupakan *password* yang tekah dibuat sebelumnya. *Field* ini menggunakan tipe data VARCHAR dan panjang data 100 karakter. |
| 8 | created\_at | TIMESTAMP | *Field* ini berisi keteranga waktu akun profil *admin* telah berhasil didaftarkan (dibentuk) pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah TIMESTAMP. |
| 9 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika terjadi pembaharuan profil akun *admin* pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah TIMESTAMP. |

Tabel 3. 3 merupakan kamus data entitas *admins* yang memuat atribut atau *field* yang ada pada entitas *admins* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *admins*, *field* id dipakai sebagai *primary key*.

Tabel 3. 4 Kamus Data *Admin\_Notifications*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari notifikasi yang diterima *admin* pada aplikasi *e- commerce* dan digunakan sebagai *primary key* dari entitas *admin\_notifications*. Tipe data dari *field* ini adalah INT. |
| 2 | type | VARCHAR[100] | *Field* ini berisi tipe notifikasi yang diterima *admin* pada aplikasi *e- commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 3 | notifiable\_type | VARCHAR[100] | *Field* ini berisi tipe *admin* notifikasi pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 4 | notifiable\_id | INT | *Field* ini berisi id *admin* yang menerima  Notifikasi pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *admins*. |
| 5 | data | TEXT | *Field* ini berisi teks yang akan ditampilkan pada notifikasi yang diterima. Tipe data yang digunakan pada *field* ini adalah TEXT. |
| 6 | read\_at | TIMESTAMP | *Field* ini berisi keterangan waktu saat notifikasi telah dibaca oleh penerima (*admin*) dengan tipe data TIMESTAMP. |
| 7 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu pada saat notifikasi penerima (*admin*) dibuat. *Field* ini dibuat dengan tipe data TIMESTAMP. |
| 8 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika notifikasi yang diterima oleh *admin* telah diperbaharui dengan tipe data TIMESTAMP. |

Tabel 3. 4 merupakan kamus data entitas *admin\_notifications* yang memuat atribut atau *field* yang ada pada entitas *admin\_notifications* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *admin\_notifications*, *field* id dipakai sebagai *primary key*.

Tabel 3. 5 Kamus Data *Products*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari produk yang terdaftar pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *products.* Tipe datadari *field* ini adalah INT*.* |
| 2 | product\_name | VARCHAR[100] | *Field* ini berisi data nama produk yang terdaftar pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 3 | price | INT | *Field* ini berisi data harga dari produk yang terdaftar pada aplikasi *e- commerce* dengan tipe data INT. |
| 4 | description | TEXT | *Field* ini berisi data deskripsi produk yang terdaftar pada aplikasi *e- commerce*. Tipe data dari *field* ini adalah TEXT. |
| 5 | product\_rate | DOUBLE | *Field* ini berisi data *rating* produk yang  terdaftar pada aplikasi *e-commerece.* Tipe data yang digunakan pada *field* ini adalah DOUBLE. |
| 6 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika sebuah produk didaftarkan pada aplikasi *e-commerce*. Tipe data yang digunakan pada *field* ini adalah TIMESTAMP. |
| 7 | updated\_at | TIMESTAMP | Field ini berisi keterangan waktu ketika sebuah produk di-*update* datanya pada aplikasi *e- commerce* dengan tipe data TIMESTAMP. |
| 8 | stock | INT | *Field* ini berisi data tentang stok dari barang atau produk yang terdaftar pada aplikasi *e- commerce* dengan tipe data INT. |
| 9 | weight | INT | *Field* ini berisi data tentang berat produk yang terdaftar pada aplikasi *e-commerce* dengan tipe data INT. |

Tabel 3. 5 merupakan kamus data entitas *products* yang memuat atribut atau *field* yang ada pada entitas *products* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *products*, *field* id dipakai sebagai *primary key*.

Tabel 3. 6 Kamus Data *Product\_Reviews*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari *review* produk pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *product\_reviews*. Tipe data dari *field* ini adalah INT. |
| 2 | product\_id | INT | *Field* ini berisi id produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* id pada entitas *products*. |
| 3 | user\_id | INT | *Field* ini berisi id *user* yang memberikan ulasan produk atau *review* pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *users*. |
| 4 | content | TEXT | *Field* ini berisi data konten ulasan produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah TEXT. |
| 5 | rate | INT | *Field* ini berisi data *rating* yang diberikan pengguna terhadap sebuah produk yang terdaftar pada aplikasi *e- commerce*. Tipe data yang digunakan pada *field* ini adalah INT. |
| 6 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika ulasan sebuah produk dibuat pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 7 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika ulasan sebuah produk di-*update* pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 6 merupakan kamus data entitas *product\_reviews* yang memuat atribut atau *field* yang ada pada entitas *product\_reviews* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *product\_reviews*, *field* id dipakai sebagai *primary key*.

Tabel 3. 7 Kamus Data *Product\_Images*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari gambar produk pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *product\_images*. Tipe data dari *field* ini adalah INT. |
| 2 | product\_id | INT | *Field* ini berisi id produk pada aplikasi *e-commerce.* Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *products*. |
| 3 | image\_name | VARCHAR[100] | *Field* ini berisi data URL dari gambar produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 4 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika gambar sebuah produk ditambahkan pada aplikasi *e-commerece* dengan tipe data TIMESTAMP. |
| 5 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika gambar sebuah produk di-*update* pada aplikasi *e- commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 7 merupakan kamus data entitas *product\_images* yang memuat atribut atau *field* yang ada pada entitas *product\_images* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *product\_images*, *field* id dipakai sebagai *primary key*.

Tabel 3. 8 Kamus Data *Product\_Categories*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id dari kategori produk pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *product\_categories*. Tipe data dari *field* ini adalah INT. |
| 2 | category\_name | VARCHAR[100] | *Field* ini berisi data nama kategori produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang panjang data 100 karakter. |
| 3 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika kategori sebuah produk dibuat pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 4 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika kategori sebuah produk di-*update* pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 8 merupakan kamus data entitas *product\_categories* yang memuat atribut atau *field* yang ada pada entitas *product\_categories* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *product\_categories*, *field* id dipakai sebagai *primary key*.

Tabel 3. 9 Kamus Data *Products\_Category\_Details*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id detail kategori produk pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *products\_category\_details*. Tipe data dari *field* ini adalah INT. |
| 2 | product\_id | INT | *Field* ini berisi id produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *products*. |
| 3 | category\_id | INT | *Field* ini berisi id kategori pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *product\_categories*. |
| 4 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika detail kategori sebuah produk dibuat pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 5 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika detail kategori sebuah produk di-*update* pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 9 merupakan kamus data entitas *products\_category\_details* yang memuat atribut atau *field* yang ada pada entitas *products\_category\_details* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *products\_category\_details*, *field* id dipakai sebagai *primary key*.

Tabel 3. 10 Kamus Data *Carts*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id keranjang berisi produk pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *carts*. Tipe data dari *field* ini adalah INT. |
| 2 | product\_id | INT | *Field* ini berisi id produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *products*. |
| 3 | user\_id | INT | *Field* ini berisi id pengguna yang terdaftar pada aplikasi *e-commerce*. Tipe data dari field ini adalah INT. Field ini merupakan *foreign key* dari atribut id pada entitas *users*. |
| 4 | qty | INT | *Field* ini berisi kuantitas produk yang dimasukkan ke dalam keranjang pada aplikasi *e-commeree* dengan tipe data INT. |
| 5 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika sebuah produk ditambahkan ke dalam keranjang pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 6 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika keranjang di-*update* isinya pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 7 | status | ENUM | *Field* ini berisi status produk yang dimasukkan ke dalam keranjang dengan tipe data ENUM dengan pilihan ‘CHECKEDOUT’, ’NOTYET’, dan ‘CANCELLED’ |

Tabel 3. 10 merupakan kamus data entitas *carts* yang memuat atribut atau *field* yang ada pada entitas *carts* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *carts*, *field* id dipakai sebagai *primary key*.

Tabel 3. 11 Kamus Data *Couriers*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisI data Id kurir pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *couriers*. Tipe data dari *field* ini adalah INT. |
| 2 | courier | VARCHAR[100] | *Field* ini berisi data kurir pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang panjang data 100 karakter. |
| 3 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika data kurir ditambahkan ke dalam keranjang pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 4 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika data kurir diperbaharui pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 11 merupakan kamus data entitas *couriers* yang memuat atribut atau *field* yang ada pada entitas *couriers* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *couriers*, *field* id dipakai sebagai *primary key*.

Tabel 3. 12 Kamus Data *Response*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id respon atas *review* produk pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *responses*. Tipe data dari *field* ini adalah INT. |
| 2 | review\_id | INT | *Field* ini berisi id ulasan produk pada aplikasi *e-commerce.* Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *product\_reviews*. |
| 3 | admin\_id | INT | *Field* ini berisi id *admin* yang merespon *review* produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *admins*. |
| 4 | content | TEXT | *Field* ini berisi teks dari respon atas ulasan produk pada aplikasi *e-commerece* dengan tipe data TEXT. |
| 5 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika respon atas ulasan produk ditambahkan pada aplikasi *e- commerce* dengan tipe data TIMESTAMP. |
| 6 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika respon atas ulasan produk diubah pada aplikasi *e- commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 12 merupakan kamus data entitas *responses* yang memuat atribut atau *field* yang ada pada entitas *responses* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *responses*, *field* id dipakai sebagai *primary key*.

Tabel 3. 13 Kamus Data *Discounts*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data diskon atau potongan harga pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *discounts*. Tipe data dari *field* ini adalah INT. |
| 2 | id\_products | INT | *Field* ini berisi id produk yang mendapat potongan harga pada aplikasi *e-commerce.* Tipe data dari *field* ini adalah INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *products*. |
| 3 | percentage | INT | *Field* ini berisi besaran potongan harga sebuah produk pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. |
| 4 | start | DATE | *Field* ini berisi tanggal dimulainya pemberlakuan potongan harga pada suatu produk pada aplikasi *e-commerece* dengan tipe data DATE. |
| 5 | end | DATE | *Field* ini berisi tanggal berakhirnya pemberlakuan potongan harga pada suatu produk pada aplikasi *e-commerce* dengan tipe data DATE. |
| 6 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika potongan harga sebuah produk dibuat pada aplikasi *e- commerce* dengan tipe data TIMESTAMP. |
| 7 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika potongan harga sebuah produk diperbaharui pada aplikasi *e- commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 13 merupakan kamus data entitas *discounts* yang memuat atribut atau *field* yang ada pada entitas *discounts* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *discounts*, *field* id dipakai sebagai *primary key*.

Tabel 3. 14 Kamus Data *Transactions*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id transaksi yang terjadi pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *transactions*. Tipe data dari *field* ini adalah INT. |
| 2 | timeout | DATETIME | *Field* ini berisi keterangan waktu *timeout* dari suatu transaksi dari *user* pada aplikasi *e-commerce.* Tipe data dari *field* ini adalah DATETIME. |
| 3 | address | VARCHAR[100] | *Field* ini berisi alamat pengiriman barang pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 4 | regency | VARCHAR[100] | *Field* ini berisi kabupaten/kota dari alamat pengiriman barang pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 5 | province | VARCHAR[100] | *Field* ini berisi provinsi dari alamat pengiriman barang pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah VARCHAR dengan panjang data 100 karakter. |
| 6 | total | DOUBLE | *Field* ini berisi total harga produk yang dibeli pada suatu transaksi pada aplikasi *e- commerce* dengan tipe data DOUBLE. |
| 7 | shipping\_cost | DOUBLE | *Field* ini berisi harga ongkos pengiriman untuk produk yang dibeli pelanggan pada aplikasi *e- commerce* dengan tipe data DOUBLE. |
| 8 | sub\_total | DOUBLE | *Field* ini berisi total harga yang sudah termasuk ongkos pengiriman pada suatu transaksi pada aplikasi *e- commerce* dengan tipe data DOUBLE. |
| 9 | user\_id | INT | *Field* ini berisi id dari pengguna yang melakukan transaksi. *Field* ini merupakan *foreign key* dari atribut id di entitas *users*. |
| 10 | courier\_id | INT | *Field* ini berisi id dari kurir yang akan mengirimkan produk pada suatu transaksi. *Field* ini merupakan *foreign key* dari atribut id di entitas *couriers*. |
| 11 | proof\_of\_payment | VARCHAR[100] | *Field* ini berisi bukti pembayaran dari transaksi yang dilakukan pada aplikasi *e-commerce* dengan tipe data VARCHAR dan panjang data 100 karakter. |
| 12 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika sebuah transaksi dibuat pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 13 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika sebuah transaksi diubah pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 14 | status | ENUM | *Field* ini berisi status transaksi pada aplikasi *e-commerce* dengan tipe data ENUM dengan pilihan ‘UNVERIFIED’, ’VERIFIED’, ’DELIVERED’, ’SUCCESS’, ‘EXPIRED’, atau ‘CANCELLED’. |

Tabel 3. 14 merupakan kamus data entitas *transactions* yang memuat atribut atau *field* yang ada pada entitas *transactions* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *transactions*, *field* id dipakai sebagai *primary key*.

Tabel 3. 15 Kamus Data *Transaction\_Details*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Field** | **Type** | **Keterangan** |
| 1 | id | INT | *Field* ini berisi data id detail transaksi yang terjadi pada aplikasi *e-commerce* dan digunakan sebagai *primary key* dari entitas *transaction\_details*. Tipe data dari *field* ini adalah INT. |
| 2 | transaction\_id | INT | *Field* ini berisi id dari transaksi pada aplikasi *e-commerce* dengan tipe data INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *transactions*. |
| 3 | product\_id | INT | *Field* ini berisi id produk yang dibeli pada aplikasi *e-commerce* dengan tipe data INT. *Field* ini merupakan *foreign key* dari atribut id pada entitas *products*. |
| 4 | qty | INT | *Field* ini berisi kuantitas barang yang dibeli pada aplikasi *e-commerce* dengan tipe data INT. |
| 5 | discount | INT | *Field* ini berisi diskon barang yang dibeli pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. |
| 6 | selling\_price | INT | *Field* ini berisi harga barang yang dibeli pada aplikasi *e-commerce*. Tipe data dari *field* ini adalah INT. |
| 7 | created\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika sebuah detail transaksi ditambahkan pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |
| 8 | updated\_at | TIMESTAMP | *Field* ini berisi keterangan waktu ketika sebuah detail transaksi diubah pada aplikasi *e-commerce* dengan tipe data TIMESTAMP. |

Tabel 3. 15 merupakan kamus data entitas *transaction\_details* yang memuat atribut atau *field* yang ada pada entitas *transaction\_details* pada rancangan *database* aplikasi penjualan atau *e-commerce* berbasis web. Pada entitas *transaction\_details*, *field* id dipakai sebagai *primary key*.

# BAB IV PEMBAHASAN DAN HASIL ANALISA

Bab IV berisi pembahasan dan hasil analisa dari penelitian yang telah dilakukan per masing-masing modul praktikum. Modul pertama akan membahas tentang *multi authentication*, dilanjutnya dengan *master* dan detail data pada modul dua, transaksi di modul ketiga, dan *report* pada modul keempat. Rincian dari pembahasan hasil penelitian dapat dilihat sebagai berikut.

## 4.1 Multi Authentication

Modul pertama praktikum pemrograman internet yaitu penerapan *multi- authentication* pada sistem *web e-commerce* dimana terdapat dua jenis pengguna yaitu *admin* dan *user* (pembeli). *Multi-authentication* ini dilakukan untuk membedakan akses yang didapatkan oleh *admin* dan *user*. Proses *multi- authentication* ini dibentuk dengan membuat sebuah *guard* pada *file* auth.php terlebih dahulu. Berikut ini akan ditampilkan kode program *guard* untuk menerapkan *multi-authentication* pada sistem *e-commerce*.

|  |
| --- |
| <?php  return [  /\*  |--------------------------------------------------------------------------  | Authentication Defaults  |--------------------------------------------------------------------------  |  | This option controls the default authentication "guard" and password  | reset options for your application. You may change these defaults  | as required, but they're a perfect start for most applications.  |  \*/  'defaults' => [  'guard' => 'web',  'passwords' => 'users',  ],    /\*  |--------------------------------------------------------------------------  | Authentication Guards  |--------------------------------------------------------------------------  |  | Next, you may define every authentication guard for your application.  | Of course, a great default configuration has been defined for you  | here which uses session storage and the Eloquent user provider.  |  | All authentication drivers have a user provider. This defines how the  | users are actually retrieved out of your database or other storage  | mechanisms used by this application to persist your user's data.  |  | Supported: "session", "token"  |  \*/  'guards' => [  'web' => [  'driver' => 'session',  'provider' => 'users',  ],  'admin' => [  'driver' => 'session',  'provider' => 'admins',  ],  'api' => [  'driver' => 'token',  'provider' => 'users',  'hash' => false,  ],  ],  /\*  |--------------------------------------------------------------------------  | User Providers  |--------------------------------------------------------------------------  |  | All authentication drivers have a user provider. This defines how the  | users are actually retrieved out of your database or other storage  | mechanisms used by this application to persist your user's data.  |  | If you have multiple user tables or models you may configure multiple  | sources which represent each model / table. These sources may then  | be assigned to any extra authentication guards you have defined.  |  | Supported: "database", "eloquent"  |  \*/  'providers' => [  'users' => [  'driver' => 'eloquent',  'model' => App\Models\User::class,  ],  'admins' => [  'driver' => 'eloquent',  'model' => App\Models\Admin::class,  ],  // 'users' => [  // 'driver' => 'database',  // 'table' => 'users',  // ],  ],  /\*  |--------------------------------------------------------------------------  | Resetting Passwords  |--------------------------------------------------------------------------  |  | You may specify multiple password reset configurations if you have more  | than one user table or model in the application and you want to have  | separate password reset settings based on the specific user types.  |  | The expire time is the number of minutes that the reset token should be  | considered valid. This security feature keeps tokens short-lived so  | they have less time to be guessed. You may change this as needed.  |  \*/  'passwords' => [  'users' => [  'provider' => 'users',  'table' => 'password\_resets',  'expire' => 60,  'throttle' => 60,  ],  ],  /\*  |--------------------------------------------------------------------------  | Password Confirmation Timeout  |--------------------------------------------------------------------------  |  | Here you may define the amount of seconds before a password confirmation  | times out and the user is prompted to re-enter their password via the  | confirmation screen. By default, the timeout lasts for three hours.  |  \*/  'password\_timeout' => 10800,  ]; |

Kode Program 4. 1 *Source Code* auth.php

Kode program 4.1 merupakan *source code* dari file auth.php dimana dalam *file* tersebut akan diatur *authentication guard* untuk menerapkan sistem *multi-authentication* pada sistem *e-commerce*. Berikut ini akan dijelaskan *source code* serta tampilan *login* dan *register* untuk *user* dan *admin* dalam penerapan konsep *multi-authentication* pada sistem *e-commerce*.

### 4.1.1 Login Admin

Pada sistem *e-commerce* akan dilakukan pembagian hak akses atas pengguna *e-commerce* yang berstatus *admin* dan *user*. Berikut merupakan kode program dan tampilan yang memuat sistem *login* bagi pengguna *e-commerce* yang berstatus sebagai *admin*.

#### 4.1.1.1 Source Code

*Login* oleh pengguna dengan berstatus *admin* dan pemberian hak akses kepada pengguna dengan status *admin*, diterapkan dengan menggunakan konsep MVC. Berikut ini merupakan kode program dalam implementasi *login admin* dan pemberian hak akses pada *admin*.

1. Model

Model admin.php merupakan file model yang memiliki fungsi untuk mengelola proses manipulasi data yang terkait dengan data *admin* yang terdaftar pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  use Illuminate\Foundation\Auth\User as Authenticable;  use Illuminate\Notifications\Notifiable;  class Admin extends Authenticable  {  use Notifiable;  protected $guard = 'admin';  protected $fillable = [  'name', 'email', 'username', 'password', 'email\_verfied\_at', 'profile\_image', 'phone'  ];  protected $hidden = ['password'];  public function getImageAttribute()  {  return $this->profile\_image ? asset('storage/' . $this->profile\_image) : asset('assets/images/default-user.jpg');  }  public function notifications()  {  return $this->morphMany(AdminNotification::class, 'notifiable')->orderBy('created\_at', 'desc');  }  public function unreadNotifications()  {  return $this->morphMany(AdminNotification::class, 'notifiable')->where('read\_at',null)  ->orderBy('created\_at', 'desc');  }  } |

Kode Program 4. 2 *Source Code* admin.php

Kode program 4. 2 merupakan kode program dari model admin.php yang merupakan modelyang memiliki fungsi untuk mengelola proses manipulasi data yang terkait dengan data *admin* yang terdaftar pada sistem *e-commerce*.

1. View

*View login admin* merupakan bagian *view* yang berisi kode program tampilan dari halaman *login* khusus untuk hak akses *admin* pada sistem *e- commerce*. Berikut merupakan kode program yang digunakan untuk membentuk tampilan halaman *login admin*.

|  |
| --- |
| @extends('layouts.app')  @section('content')  <!-- Navbar-->  <header class="header">  <nav class="navbar navbar-expand-lg navbar-light py-3">  <div class="container">  <!-- Navbar Brand -->  <a href="#" class="navbar-brand">    </a>  </div>  </nav>  </header>  <div class="container">  <div class="row align-items-center" style="padding-top: 10%">    <!-- Registeration Form -->  <div class="container col-8">  <h1 class=" pb-3">Login Admin</h1>  <form method="POST" action="">  @csrf  <div class="row">  <!-- Email Address -->    <!-- <label>Username</label> -->  <div class="input-group col-lg-12 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-user-o text-muted"></i>  </span>  </div>  <input id="username" type="text" placeholder="Username" class="form-control" name="username" value="{{ old('username') }}"  required autocomplete="username" autofocus>    @if(Session::get('error'))  <span class="invalid-feedback d-block" role="alert">  <strong>{{ Session::get('error') }}</strong>  </span>  @endif  </div>  <!-- Password -->  <div class="input-group col-lg-12 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-lock text-muted"></i>  </span>  </div>  <input id="password" type="password" placeholder="Password" class="form-control" name="password" required  autocomplete="current-password">    <div class="bar"></div>  @error('password')  <span class="invalid-feedback d-block" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  <button type="submit" class="btn btn-primary btn-block py-2">  <span class="font-weight-bold">Login</span>  </button>  </div>  </div>  </form>  </div>  <!-- For Demo Purpose -->    </div>  </div>  @endsection |

Kode Program 4. 3 *Source Code* login.blade.php

Kode program 4. 3 merupakan kode program dari model login.blade.php yang merupakan bagian *view* yang berisi kode program yang membangun tampilan dari halaman *login* khusus untuk hak akses *admin* pada sistem *e*-*commerce*.

1. Controller

LoginController merupakan *controller* yang akan melakukan *authentication* akun-akun *admin* dan *user* saat melakukan proses *login* ke dalam sistem. *Login* tersebut disajikan pada bagian modelauth.admin\_login dan auth.login pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  use Illuminate\Foundation\Auth\User as Authenticable;  use Illuminate\Notifications\Notifiable;  class Admin extends Authenticable  {  use Notifiable;  protected $guard = 'admin';  protected $fillable = [  'name', 'email', 'username', 'password', 'email\_verfied\_at', 'profile\_image', 'phone'  ];  protected $hidden = ['password'];  public function getImageAttribute()  {  return $this->profile\_image ? asset('storage/' . $this->profile\_image) : asset('assets/images/default-user.jpg');  }  public function notifications()  {  return $this->morphMany(AdminNotification::class, 'notifiable')->orderBy('created\_at', 'desc');  }  public function unreadNotifications()  {  return $this->morphMany(AdminNotification::class, 'notifiable')->where('read\_at',null)  ->orderBy('created\_at', 'desc');  }  } |

Kode Program 4. 4 *Source Code Login Controller*

Kode program 4. 4 merupakan kode program dari LoginController. *Controller* ini berfungsi untuk mengatur bagaimana akses yang akan diberikan oleh sistem nantinya kepada pengguna, apakah dia mendapat akses *admin* atau *user*.

#### 4.1.1.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi *login admin* dan pemberian hak akses kepada *admin*. Berikut ini akan ditampilkan tampilan dari sistem *login admin* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 1 Tampilan Halaman *Login*

Gambar 4. 1 merupakan tampilan halaman *login admin* dimana pengguna berstatus *admin* yang telah terdaftar pada sistem *e-commerce* memasukkan *username* terdaftar dan *password* untuk dapat mengakses halaman *admin*.

|  |
| --- |
|  |

Gambar 4. 2 Tampilan Halaman *Admin*

Gambar 4. 2 merupakan tampilan halaman admin berisi sejumlah akses yang hanya diberikan pada pengguna yang berstatus atau terdaftar sebagai *admin* pada sistem *e-commerce*.

### 4.1.2 Login User

Pada sistem *e-commerce* akan dilakukan pembagian hak akses atas pengguna *e-commerce* yang berstatus *admin* dan *user*. Berikut merupakan kode program dan tampilan yang memuat sistem *login* bagi pengguna *e-commerce* yang berstatus sebagai *user* biasa.

#### 4.1.3.1 Source Code

*Login* oleh pengguna dengan berstatus *user* dan pemberian hak akses kepada pengguna dengan status *user*, diterapkan dengan menggunakan konsep MVC. Berikut ini merupakan kode program dalam implementasi *login user* dan pemberian hak akses pada *user*.

a Model

Model *User* merupakan model yang memiliki fungsi untuk mengelola proses manipulasi informasi yang terkait dengan data dari *user* yang terdaftar pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Contracts\Auth\MustVerifyEmail;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Foundation\Auth\User as Authenticatable;  use Illuminate\Notifications\Notifiable;  class User extends Authenticatable implements MustVerifyEmail  {  use HasFactory, Notifiable;  /\*\*  \* The attributes that are mass assignable.  \*  \* @var array  \*/  protected $fillable = [  'name',  'email',  'password',  'profile\_image'  ];  /\*\*  \* The attributes that should be hidden for arrays.  \*  \* @var array  \*/  protected $hidden = [  'password',  'remember\_token',  ];  /\*\*  \* The attributes that should be cast to native types.  \*  \* @var array  \*/  protected $casts = [  'email\_verified\_at' => 'datetime',  ];  public function Reviews(){  return $this->hasMany(ReviewProducts::class,'user\_id');  }  public function transaction(){  return $this->hasMany(Transactions::class,'user\_id');  }  public function getImageAttribute()  {  return $this->profile\_image ? asset('storage/' . $this->profile\_image) : asset('assets/images/default-user.jpg');  }  public function notifications()  {  return $this->morphMany(UserNotification::class, 'notifiable')  ->orderBy('created\_at', 'desc');  }  public function unreadNotifications()  {  return $this->morphMany(UserNotification::class, 'notifiable')->where('read\_at',null)  ->orderBy('created\_at', 'desc');  }    } |

Kode Program 4. 5 *Source Code Login User*

Kode program 4. 5 merupakan kode program dari model user yaitu model yang akan mengelola dan memanipulasi data terkait *user* yang terdaftar pada sistem *e-commerce*.

b. View

*View* login merupakan bagian *view* yang akan menampilkan halaman *login* khusus untuk hak akses *user* dari sistem *e-commerce*. Berikut ini merupakan kode program dari *view* login.

|  |
| --- |
| @extends('layouts.app')  @section('content')  <!-- Navbar-->  <header class="header">  <nav class="navbar navbar-expand-lg navbar-light py-3">  <div class="container">  <!-- Navbar Brand -->  <a href="#" class="navbar-brand">    </a>  </div>  </nav>  </header>  <div class="container">  <div class="row align-items-center" style="padding-top: 10%">    <!-- Registeration Form -->  <div class="container col-8">  <h1 class="center pb-3">Login user</h1>  <form method="POST" action="{{ route('login') }}">  @csrf  <div class="row">  <!-- Email Address -->  <div class="input-group col-lg-12 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-envelope text-muted"></i>  </span>  </div>  <input id="email" placeholder="Email Address" type="email" class="form-control @error('email') is-invalid @enderror bg-white border-left-0 border-md" name="email" value="{{ old('email') }}" required autocomplete="email" autofocus>  @error('email')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  <!-- Password -->  <div class="input-group col-lg-12 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-lock text-muted"></i>  </span>  </div>  <input id="password" type="password" placeholder="Password" class="form-control @error('password') is-invalid @enderror bg-white border-left-0 border-md" name="password" required autocomplete="current-password">  @error('password')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  <!-- Submit Button -->  <div class="form-group col-lg-12 mx-auto mb-0">  <p class="font-weight-bolder text-center"><u><a class="text-danger " href="/password/reset">Forgot Password</a></u></p>  <button type="submit" class="btn btn-primary btn-block py-2">  <span class="font-weight-bold">Login</span>  </button>  </div>  <!-- Divider Text -->  <div class="form-group col-lg-12 mx-auto d-flex align-items-center my-4">  <div class="border-bottom w-100 ml-5"></div>  <span class="px-2 small text-muted font-weight-bold text-muted">OR</span>  <div class="border-bottom w-100 mr-5"></div>  </div>  <!-- don't have an account -->  <div class="text-center w-100">  <p class="text-muted font-weight-bold">don't have an account? <a href="/register" class="text-danger ml-2">Register here</a></p>  </div>  </div>  </form>  </div>  <!-- For Demo Purpose -->    </div>  </div>  @endsection |

Kode Program 4. 6 *Source Code View Login*

Kode program 4. 6 merupakan kode program dari *view* login yaitu bagian *view* yang akan menampilkan halaman *login* khusus untuk hak akses *user* dari sistem *e-commerce*.

C Controller

LoginController merupakan *controller* yang akan melakukan *authentication* akun-akun *admin* dan *user* saat melakukan proses *login* ke dalam sistem. *Login* tersebut disajikan pada bagian *view* auth.admin\_login dan auth.login pada sistem.

|  |
| --- |
| <?php  namespace App\Http\Controllers\Auth;  use App\Http\Controllers\Controller;  use App\Providers\RouteServiceProvider;  use App\Models\User;  use Illuminate\Foundation\Auth\RegistersUsers;  use Illuminate\Support\Facades\Hash;  use Illuminate\Support\Facades\Validator;  class RegisterController extends Controller  {  /\*  |--------------------------------------------------------------------------  | Register Controller  |--------------------------------------------------------------------------  |  | This controller handles the registration of new users as well as their  | validation and creation. By default this controller uses a trait to  | provide this functionality without requiring any additional code.  |  \*/  use RegistersUsers;  /\*\*  \* Where to redirect users after registration.  \*  \* @var string  \*/  protected $redirectTo = RouteServiceProvider::HOME;  /\*\*  \* Create a new controller instance.  \*  \* @return void  \*/  public function \_\_construct()  {  $this->middleware('guest');  }  /\*\*  \* Get a validator for an incoming registration request.  \*  \* @param array $data  \* @return \Illuminate\Contracts\Validation\Validator  \*/  protected function validator(array $data)  {  return Validator::make($data, [  'name' => ['required', 'string', 'max:255'],  'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],  'password' => ['required', 'string', 'min:8', 'confirmed'],  ]);  }  /\*\*  \* Create a new user instance after a valid registration.  \*  \* @param array $data  \* @return \App\Models\User  \*/  protected function create(array $data)  {  return User::create([  'name' => $data['name'],  'email' => $data['email'],  'password' => Hash::make($data['password']),  ]);  }  } |

Kode Program 4. 7 *Source Code Login Controller*

Kode program 4. 7 merupakan kode program dari LoginController. *Controller* ini berfungsi untuk mengatur bagaimana akses yang akan diberikan oleh sistem nantinya kepada pengguna, apakah dia mendapat akses *admin* atau *user*.

#### 4.1.3.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi *login user* dan pemberian hak akses kepada *user*. Berikut ini akan ditampilkan tampilan dari sistem *login user* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 3 Tampilan Halaman *Login User*

Gambar 4. 3 merupakan tampilan halaman *login user* dimana *user* terlebih dahulu harus terdaftar dalam sistem dengan melakukan *register* lalu memasukkan alamat *e-mail* yang telah terdaftar dan *password* untuk melakukan *login*.

|  |
| --- |
|  |

Gambar 4. 4 Tampilan Halaman Utama

Gambar 4. 4 merupakan tampilan halaman utama dari sistem *e-commerce*. Setelah melakukan *login*, pengguna akan langsung diarahkan menuju halaman utama dari sistem toko *online* ini.

### 4.1.3 Register User

Proses *register user* tentu dibutuhkan untuk mendaftarkan pengguna yang akan mendapatkan hak akses ke dalam halaman *user*. Proses ini tentu harus dibedakan dengan proses registrasi admin. Berikut merupakan kode program dan tampilan untuk *register user*.

#### 4.1.3.1 Source Code

*Register* pengguna dengan berstatus *admin* diterapkan dengan menggunakan konsep MVC. Berikut ini merupakan kode program dalam implementasi *register admin* dalam rancangan sistem *e-commerce*.

a Model

Model yang digunakan untuk *fitur* ini masih sama dengan model yang digunakan pada *login user*. Model User merupakan model yang memiliki fungsi untuk mengelola proses manipulasi informasi yang terkait dengan data dari *user* yang terdaftar pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Contracts\Auth\MustVerifyEmail;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Foundation\Auth\User as Authenticatable;  use Illuminate\Notifications\Notifiable;  class User extends Authenticatable implements MustVerifyEmail  {  use HasFactory, Notifiable;  /\*\*  \* The attributes that are mass assignable.  \*  \* @var array  \*/  protected $fillable = [  'name',  'email',  'password',  'profile\_image'  ];  /\*\*  \* The attributes that should be hidden for arrays.  \*  \* @var array  \*/  protected $hidden = [  'password',  'remember\_token',  ];  /\*\*  \* The attributes that should be cast to native types.  \*  \* @var array  \*/  protected $casts = [  'email\_verified\_at' => 'datetime',  ];  public function Reviews(){  return $this->hasMany(ReviewProducts::class,'user\_id');  }  public function transaction(){  return $this->hasMany(Transactions::class,'user\_id');  }  public function getImageAttribute()  {  return $this->profile\_image ? asset('storage/' . $this->profile\_image) : asset('assets/images/default-user.jpg');  }  public function notifications()  {  return $this->morphMany(UserNotification::class, 'notifiable')  ->orderBy('created\_at', 'desc');  }  public function unreadNotifications()  {  return $this->morphMany(UserNotification::class, 'notifiable')->where('read\_at',null)  ->orderBy('created\_at', 'desc');  }    } |

Kode Program 4. 8 *Source Code Register User*

Kode program 4. 8 merupakan kode program dari model *user* yaitu model yang akan mengelola dan memanipulasi data terkait *user* yang terdaftar pada sistem *e-commerce*.

b View

*View* register merupakan bagian *view* yang menampilkan halaman yang berisi beberapa *field* yang memerlukan *input* dari *user* untuk melanjutkan proses pendaftaran ke dalam sistem *e-commerce*

|  |
| --- |
| {{-- @extends('layouts.app')  @section('content')  <div class="container">  <div class="row justify-content-center">  <div class="col-md-8">  <div class="card">  <div class="card-header">{{ \_\_('Register') }}</div>  <div class="card-body">  <form method="POST" action="{{ route('register') }}">  @csrf  <div class="form-group row">  <label for="name" class="col-md-4 col-form-label text-md-right">{{ \_\_('Name') }}</label>  <div class="col-md-6">  <input id="name" type="text" class="form-control @error('name') is-invalid @enderror" name="name" value="{{ old('name') }}" required autocomplete="name" autofocus>  @error('name')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  </div>  <div class="form-group row">  <label for="email" class="col-md-4 col-form-label text-md-right">{{ \_\_('E-Mail Address') }}</label>  <div class="col-md-6">  <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}" required autocomplete="email">  @error('email')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  </div>  <div class="form-group row">  <label for="password" class="col-md-4 col-form-label text-md-right">{{ \_\_('Password') }}</label>  <div class="col-md-6">  <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password" required autocomplete="new-password">  @error('password')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  </div>  <div class="form-group row">  <label for="password-confirm" class="col-md-4 col-form-label text-md-right">{{ \_\_('Confirm Password') }}</label>  <div class="col-md-6">  <input id="password-confirm" type="password" class="form-control" name="password\_confirmation" required autocomplete="new-password">  </div>  </div>  <div class="form-group row mb-0">  <div class="col-md-6 offset-md-4">  <button type="submit" class="btn btn-primary">  {{ \_\_('Register') }}  </button>  </div>  </div>  </form>  </div>  </div>  </div>  </div>  </div>  @endsection --}}  @extends('layouts.app')  @section('content')  <!-- Navbar-->  <header class="header">  <nav class="navbar navbar-expand-lg navbar-light py-3">  <div class="container">  <!-- Navbar Brand -->  <a href="#" class="navbar-brand">    </a>  </div>  </nav>  </header>  <div class="container">  <div class="row align-items-center" style="padding-top: 10%">  <!-- For Demo Purpose -->  <div class="container">  <!-- <img src="/assets/images/vector\_register-01.png" alt="" class="img-fluid mb-3 d-none d-md-block"> -->  <!-- <h1>Create an Account</h1> -->      </div>  <!-- Registeration Form -->  <div class="container col-8">  <h1 class=" pb-3">Register User</h1>  <form method="POST" action="{{ route('register') }}">  @csrf  <div class="row">  <!-- First Name -->  <div class="input-group col-lg-12 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-user text-muted"></i>  </span>  </div>  <input id="name" placeholder="Input Name" type="text" class="form-control @error('name') is-invalid @enderror bg-white border-left-0 border-md" name="name" value="{{ old('name') }}" required autocomplete="name" autofocus>  @error('name')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  <!-- Email Address -->  <div class="input-group col-lg-12 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-envelope text-muted"></i>  </span>  </div>  <input id="email" placeholder="Input Email" type="email" class="form-control @error('email') is-invalid @enderror bg-white border-left-0 border-md" name="email" value="{{ old('email') }}" required autocomplete="email">  @error('email')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror    </div>  <!-- Password -->  <div class="input-group col-lg-6 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-lock text-muted"></i>  </span>  </div>  <input id="password" type="password" placeholder="Input Password" class="form-control @error('password') is-invalid @enderror bg-white border-left-0 border-md" name="password" required autocomplete="new-password">  @error('password')  <span class="invalid-feedback" role="alert">  <strong>{{ $message }}</strong>  </span>  @enderror  </div>  <!-- Password Confirmation -->  <div class="input-group col-lg-6 mb-4">  <div class="input-group-prepend">  <span class="input-group-text bg-white px-4 border-md border-right-0">  <i class="fa fa-lock text-muted"></i>  </span>  </div>  <input id="password-confirm" placeholder="Confirm Password" type="password" class="form-control bg-white border-left-0 border-md" name="password\_confirmation" required autocomplete="new-password">    </div>  <!-- Submit Button -->  <div class="form-group col-lg-12 mx-auto mb-0">  <button type="submit" class="btn btn-primary btn-block py-2">  <span class="font-weight-bold">Create your account</span>  </button>  </div>  <!-- Divider Text -->  <div class="form-group col-lg-12 mx-auto d-flex align-items-center my-4">  <div class="border-bottom w-100 ml-5"></div>  <span class="px-2 small text-muted font-weight-bold text-muted">OR</span>  <div class="border-bottom w-100 mr-5"></div>  </div>    <!-- Already Registered -->  <div class="text-center w-100">  <p class="text-muted font-weight-bold">Already Registered? <a href="/login" class="text-danger ml-2">Login</a></p>  </div>  </div>  </form>  </div>  </div>  </div>  @endsection |

Kode Program 4. 9 *Source Code View Register*

Kode program 4. 9 merupakan kode program dari *view* register yaitu bagian *view* yang menampilkan halaman yang berisi beberapa *field* yang memerlukan *input* dari *user* untuk melanjutkan proses pendaftaran ke dalam sistem *e-commerce*.

c Controller

RegisterController merupakan bagian *controller* yang dapat mengatur proses pendaftaran akun *user* saat proses *register* pada sistem. Proses tersebut akan ditampilkan pada bagian *view* auth.register pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Http\Controllers\Auth;  use App\Http\Controllers\Controller;  use App\Providers\RouteServiceProvider;  use App\Models\User;  use Illuminate\Foundation\Auth\RegistersUsers;  use Illuminate\Support\Facades\Hash;  use Illuminate\Support\Facades\Validator;  class RegisterController extends Controller  {  /\*  |--------------------------------------------------------------------------  | Register Controller  |--------------------------------------------------------------------------  |  | This controller handles the registration of new users as well as their  | validation and creation. By default this controller uses a trait to  | provide this functionality without requiring any additional code.  |  \*/  use RegistersUsers;  /\*\*  \* Where to redirect users after registration.  \*  \* @var string  \*/  protected $redirectTo = RouteServiceProvider::HOME;  /\*\*  \* Create a new controller instance.  \*  \* @return void  \*/  public function \_\_construct()  {  $this->middleware('guest');  }  /\*\*  \* Get a validator for an incoming registration request.  \*  \* @param array $data  \* @return \Illuminate\Contracts\Validation\Validator  \*/  protected function validator(array $data)  {  return Validator::make($data, [  'name' => ['required', 'string', 'max:255'],  'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],  'password' => ['required', 'string', 'min:8', 'confirmed'],  ]);  }  /\*\*  \* Create a new user instance after a valid registration.  \*  \* @param array $data  \* @return \App\Models\User  \*/  protected function create(array $data)  {  return User::create([  'name' => $data['name'],  'email' => $data['email'],  'password' => Hash::make($data['password']),  ]);  }  } |

Kode Program 4. 10 *Source Code Controller Register*

Kode program 4. 10 merupakan kode program dari *controller* RegisterController yaitu bagian *controller* yang dapat mengatur proses pendaftaran akun *user* saat proses *register* pada sistem. Proses tersebut akan ditampilkan pada bagian *view* auth.register pada sistem *e-commerce*.

#### 4.1.3.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi *register user* pada rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari sistem *register user* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 5 Tampilan Halaman *Register User*

Gambar 4. 5 merupakan tampilan halaman *register user*, dimana pengguna baru dapat mendaftarkan alamat e-mailnya ke dalam sistem *e-commerce* untuk melakukan *login*.

|  |
| --- |
|  |

Gambar 4. 6 Tampilan Halaman *Login User*

Gambar 4. 6 merupakan tampilan halaman *login user* dimana *user* yang telah terdaftar tadi bisa memasukkan alamat e-mail yang terdaftar dan *password* untuk dapat melakukan *login* pada sistem *e-commerce*.

|  |
| --- |
|  |

Gambar 4. 7 Tampilan Halaman Utama

Gambar 4. 7 merupakan tampilan halaman utama dari sistem *e-commerce*. Setelah melakukan *login*, *user* akan langsung diarahkan menuju halaman utama dari sistem toko *online* ini.

## 4.2 Master dan Detail Data

Modul kedua praktikum pemrograman internet yaitu penerapan CRUD untuk master dan *detail* data dimana diterapkan CRUD pada setiap master *table* dan tabel-tabel detailnya pada rancangan aplikasi penjualan berbasis *website* dengan menerapkan *middleware* yang dibuat pada modul sebelumnya untuk menentukan akses CRUD yang diberikan pada sistem terhadap pengguna.

### 4.2.1 CRUD pada Tabel Product

Berikut ini akan ditampilkan kode program dan tampilan CRUD (*create, read, update, delete*) pada tabel *product* pada sistem *e-commerce* dengen menerapkan konsep MVC.

#### 4.2.1.1 Source Code

CRUD (*create, read, update, delete*) pada tabel *product* diterapkan dengan menggunakan konsep MVC. Berikut ini merupakan kode program dalam implementasi CRUD pada tabel *product* dalam rancangan sistem *e-commerce*.

a. Model

Product merupakan model yang memiliki fungsi untuk mengelola seluruh proses manipulasi data yang terkait dengan data *product* yang terdaftar pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Products extends Model  {  use HasFactory;  protected $guarded = [];  public function product\_category\_detail(){  return $this->hasMany(ProductCategoryDetails::class,'product\_id','id');  }    public function product\_image(){  return $this->hasMany(ProductImage::class,'product\_id','id');  }    public function category(){  return $this->belongsToMany(Category::class,'product\_category\_details', 'product\_id', 'category\_id')->withPivot('id');  }  public function Transactions(){  return $this->belongsToMany(Transactions::class, 'transaction\_details', 'product\_id', 'transaction\_id')->withPivot('id');  }  public function reviews(){  return $this->hasMany(ReviewProducts::class,'product\_id');  }  public function discounts(){  return $this->hasMany(Discounts::class, 'id\_product', 'id')->orderBy('id','DESC')->limit(1);  }  } |

Kode Program 4. 11 *Source Code Product*

Kode program 4. 11 merupakan kode program dari model Product yaitu

bagian model yang memiliki fungsi untuk mengelola seluruh proses manipulasi data yang terkait dengan data *product* yang terdaftar pada sistem *e-commerce*.

b. View

*View* admin.product.home merupakan *view* pada hak akses admin yang merupakan bagian *view* untuk menampilkan halaman berisi data setiap produk yang terdaftar pada sistem *e-commerce*. Halaman ini juga berisi manajemen penuh CRUD bagi *admin* untuk mengelola data produk yang dipasarkan pada sistem *e- commerce*.

|  |
| --- |
| @extends('/admin/layouts/master',['title'=>'Product'])  @section('content')  <section role="main" class="content-body">  <header class="page-header">  <h2>Admin</h2>  <div class="right-wrapper text-right">  <ol class="breadcrumbs pr-3">  <li>  <a href="/admin">  <i class="fas fa-home"></i>  </a>  </li>  <li><span>Product</span></li>  </ol>    </div>  </header>  <!-- start: page -->  <div class="row">  <div class="col">  <section class="card">  @include('admin/layouts/notif')  <header class="card-header">  <div class="card-actions">  <a href="#" class="card-action card-action-toggle" data-card-toggle></a>  <a href="#" class="card-action card-action-dismiss" data-card-dismiss></a>  </div>  <a href="{{Route('add-product')}}" class=" btn btn-sm btn-success"><i class="fa fa-plus"></i>  Tambah Product</a>  </header>  <div class="card-body">  <table class="dataTable table table-bordered table-striped mb-0 responsive" id="datatable-default">  <thead class="table-dark">  <tr>  <th>No</th>  <th>Nama</th>  <th>Harga</th>  <th>category</th>  <th>Stock</th>  <th>Rate</th>  <th>Berat</th>  <th class="text-center">Discount</th>  <th class="text-center">Action</th>  </tr>  </thead>  <tbody>  @php  $no = 1;  @endphp  @foreach ($products as $product)  <tr>  <td>{{$no++}}</td>  <td>{{$product->product\_name}}</td>  <td>{{$product->price}}</td>  <td>  @foreach ($product->category as $category)  {{$category->category\_name}}  @endforeach  </td>  <td>{{$product->stock}}</td>  <td>  @if ($product->reviews->avg('rate'))  @for ($i = 0; $i < 5; $i++) @if (floor($product->reviews->avg('rate')) - $i >= 1)  {{--Full Start--}}  <i class="fas fa-star text-warning"> </i>  @elseif ($product->reviews->avg('rate') - $i > 0)  {{--Half Start--}}  <i class="fas fa-star-half-alt text-warning"> </i>  @else  {{--Empty Start--}}  <i class="far fa-star text-warning"> </i>  @endif  @endfor  @else  @for ($i = 0; $i < 5; $i++) <i class="far fa-star"> </i>  @endfor  @endif  </td>  <td>{{$product->weight}} g</td>  <td class="text-center">  <a href="{{Route('discounts',['product'=>$product->id])}}" class="btn btn-sm btn-success">  @if ($product->discounts->isEmpty())  0 %  @else  @foreach ($product->discounts as $discount)  @if (date('Y-m-d') >= $discount->start && date('Y-m-d') < $discount->end)  {{$discount->percentage}} %  @else  0 %  @endif  @endforeach  @endif  </a>  </td>  <td class="text-center">  <a href="{{Route('edit-product',['product'=>$product->id])}}"  class="edit btn btn-sm btn-primary"><i class="fas fa-pencil-alt"></i></a>  <a href="{{Route('show-product',['product'=>$product->id])}}"  class=" btn btn-sm btn-dark"><i class="fas fa-eye"></i></a>  <a href="#DeletemodalForm" data-id="{{$product->id}}"  data-nama="{{$product->product\_name}}"  class="trash modal-with-form btn btn-sm btn-danger"><i  class="fa fa-trash"></i></a>  </td>  </tr>  @endforeach  </tbody>  </table>  </div>  </section>  </div>  </div>  </section>  <!-- end: page -->  </section>  <footer class="card-footer">  <div class="row">  <div class="col-md-12 text-right">  <button class="btn btn-primary" type="submit">Submit</button>  <button class="btn btn-default modal-dismiss">Cancel</button>  </div>  </div>  </footer>  </form>  </section>  </div>  <div class="modal-block modal-block-primary mfp-hide" id="DeletemodalForm" tabindex="-1" role="dialog"  aria-labelledby="myModalLabel" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <h3 id="myModalLabel">Delete Confirmation</h3>  </div>  <div class="modal-body">  <p class="error-text">  <i class="fa fa-warning modal-icon"></i>  apakah anda yakin ingin menghapus item ini?  </p>  </div>  <div class="modal-footer">  <form id="formDelete" action="#" method="post">  @method('delete')  @csrf  <button class="btn btn-default modal-dismiss" data-dismiss="modal"  aria-hidden="true">Cancel</button>  <button type="submit" class="btn btn-danger">Hapus</button>  </form>  </div>  </div>  </div>  </div>  @endsection  @section('footer')  <script>  $(".dataTable").on('click','.trash', function () {  var id = $(this).data('id');  var nama = $(this).data('nama');  $('#formDelete').attr('action', '/admin/product/destroy/' + id);  });  </script>  @endsection |

Kode Program 4. 12 *Source Code View Admin Product*

Kode program 4. 12 merupakan kode program dari *view* admin.product.home yaitu *view* pada hak akses admin untuk menampilkan halaman berisi data setiap produk yang terdaftar pada sistem *e-commerce*. Halaman ini juga berisi manajemen penuh CRUD bagi admin untuk mengelola data produk yang dipasarkan pada sistem *e-commerce*.

c. Controller

*Controller* ProductController merupakan *controller* yang menghubungkan model Product, Product\_Category\_Detail, Product\_Image, Category, dan bagian view admin.product.home dan admin.product.edit pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Http\Controllers\Admin;  use App\Models\Category;  use App\Models\Products;  use Illuminate\Http\Request;  use App\Http\Controllers\Controller;  use App\Models\Carts;  use App\Models\Discounts;  use App\Models\ProductImage;  use App\Models\Response;  use App\Models\ReviewProducts;  use Illuminate\Support\Facades\Storage;  class ProductController extends Controller  {  /\*\*  \* Display a listing of the resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function index()  {  $products = Products::all();  return view('admin/product/index', compact('products'));  }  /\*\*  \* Show the form for creating a new resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function create()  {  $categorys = Category::all();  return view('admin/product/add', compact('categorys'));  }  /\*\*  \* Store a newly created resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @return \Illuminate\Http\Response  \*/  public function store(Request $request)  {  $category = $request->category\_id; // arrays of role ids  $products = new Products;  $products->product\_name = $request->product\_name;  $products->price = $request->price;  $products->stock = $request->stock;  $products->weight = $request->weight;  $products->description = $request->description;  $products->product\_rate = '0';  $products->save();  $products->category()->attach($category);  $idproduct = Products::orderBy('id', 'desc')->first()->id;  if ($idproduct) {  $files = [];  foreach ($request->file('files') as $file) {  if ($file->isValid()) {  $nama\_image = md5(now() . "\_") . $file->getClientOriginalName();  $file->storeAs("img/products\_image", $nama\_image);  $files[] = [  'product\_id' => $idproduct,  'image\_name' => $nama\_image,  'created\_at' => now(),  'updated\_at' => now(),  ];  }  }  ProductImage::insert($files);  }  return redirect(Route('product'))->with('success', 'Berhasil menambahkan Product');  }  /\*\*  \* Display the specified resource.  \*  \* @param \App\Models\Product $product  \* @return \Illuminate\Http\Response  \*/  public function show(Products $product)  {  $notificationId = request('id');  $adminUnreadNotification = auth()->user()  ->unreadNotifications  ->where('id', $notificationId)  ->first();  // dd($notificationId);  if ($adminUnreadNotification) {  $adminUnreadNotification->update(['read\_at' => now()]);  }  $responses = Response::all();  return view('/admin/product/show', compact('product', 'responses'));  }  public function discounts(Products $product)  {  $discounts = Discounts::where('id\_product', $product->id)->get();  return view('/admin/discounts/index', compact('product', 'discounts'));  }  public function discounts\_store(Request $request)  {  $discounts = $request->all();  Discounts::create($discounts);  return redirect()->back()->with('success', 'Diskon Berhasil Ditambahkan');  }  public function discounts\_destroy(Discounts $discounts)  {  $discounts->delete();  return redirect()->back()->with('error', 'Data Telah Dihapus');  }  public function discounts\_update(Discounts $discounts, Request $request)  {  $discount = $request->all();  $discounts->update($discount);  return redirect()->back()->with('info', 'Diskon Berhasil Update');  }  /\*\*  \* Show the form for editing the specified resource.  \*  \* @param \App\Models\Product $product  \* @return \Illuminate\Http\Response  \*/  public function edit(Products $product)  {  $categorys = Category::all();  return view('admin/product/edit', compact('categorys', 'product'));  }  /\*\*  \* Update the specified resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @param \App\Models\Product $product  \* @return \Illuminate\Http\Response  \*/  public function update(Request $request, Products $product)  {  $category = $request->category\_id; // arrays of role ids  Products::find($product->id)->category()->sync($category);  $products = $request->except(['category\_id']); // pengecualian request  $product->update($products);  return redirect(Route('product'))->with('info', 'Product Telah Di Update');  }  /\*\*  \* Remove the specified resource from storage.  \*  \* @param \App\Models\Product $product  \* @return \Illuminate\Http\Response  \*/  public function destroy(Products $product)  {  foreach ($product->reviews as $review) {  Response::where('review\_id', $review->id)->delete();  }  Carts::where('product\_id', $product->id)->delete();  $product->Transactions()->detach();  $product->category()->detach();  Discounts::where('id\_product', $product->id)->delete();  foreach ($product->product\_image as $image) {  Storage::delete('img/products\_image/' . $image->image\_name);  }  ProductImage::where('product\_id', $product->id)->delete();  ReviewProducts::where('product\_id', $product->id)->delete();  $product->delete();  return redirect(Route('product'))->with('error', 'Data Telah Dihapus');  }  } |

Kode Program 4. 13 *Source Product Controller*

Kode program 4. 13 merupakan kode program dari *controller* ProductController yaitu *controller* yang menghubungkan model Product, Product\_Category\_Detail, Product\_Image, Category, dan bagian *view* admin.product.home dan admin.product.edit pada sistem *e-commerce*.

#### 4.2.1.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi CRUD pada tabel *product* dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari CRUD tabel *product* pada rancangan *e- commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 8 Halaman CRUD Tabel *Product*

Gambar 4. 8 merupakan halaman CRUD untuk tabel *product* yang memuat daftar *product* yang telah terdaftar pada sebuah *e-commerce*. *Admin* dapat mendaftarkan lebih banyak *product,* mengubah detail produk, melihat detail produk, dan menghapus produk yang telah terdaftar.

|  |
| --- |
|  |

Gambar 4. 9 Halaman *Create*

Gambar 4.9 merupakan halaman dimana *admin* dapat menambahkan atau mendaftarkan produk baru pada *e-commerce* (*create*). *Admin* diminta memasukkan nama produk, harga, berat barang, deskripsi, stok barang, dan juga gambar barang di halaman berikutnya.

|  |
| --- |
|  |

Gambar 4. 10 Halaman CRUD *Product*

Gambar 4. 10 menunjukkan halaman CRUD *product* dimana pada halaman tersebut ditunjukkan bahwa produk yang tadi didaftarkan oleh admin sudah berhasil tersimpan atau terdaftar pada *e-commerce*.

|  |
| --- |
|  |

Gambar 4. 11 Halaman Detail *Product*

Gambar 4. 11 merupakan halaman dimana *admin* dapat mengubah detail dari produk yang telah terdaftar pada *e-commerce* (*update*). *Admin* dapat mengubah detail seperti nama, harga, berat barang, deskripsi barang, stok barang, maupun menambahkan gambar terkait produk tersebut.

|  |
| --- |
|  |

Gambar 4. 12 Tampilan Detail Produk yang Sudah Diubah

Gambar 4. 12 menunjukkan halaman CRUD *product* dimana pada halaman tersebut ditunjukkan bahwa detail produk yang diubah tadi telah berhasil diubah oleh *admin*.

|  |
| --- |
|  |

Gambar 4. 13 Halaman *Admin* Dapat Melihat Detail Produk

Gambar 4. 13 merupakan halaman dimana *admin* dapat melihat detail dari produk yang telah terdaftar pada *e-commerce* (*read*) seperti nama, harga, berat barang, deskripsi barang, stok barang, maupun menambahkan gambar terkait produk tersebut.

|  |
| --- |
|  |

Gambar 4. 14 Halaman Hapus *Product*

Gambar 4. 14 menunjukkan bahwa *admin* berhasil menghapus produk yang baru saja didaftarkan dengan cara menekan tombol *delete* pada halaman CRUD di sebuah *e-commerce*.

### 4.2.2 CRUD pada Tabel Courier

Berikut ini akan ditampilkan kode program CRUD (*create, read, update, delete*) pada tabel *courier* pada sistem *e-commerce* dengen menerapkan konsep MVC.

#### 4.2.2.1 Source Code

CRUD (*create, read, update, delete*) pada tabel *courier* diterapkan dengan menggunakan konsep MVC. Berikut ini merupakan kode program dalam implementasi CRUD pada tabel *courier* dalam rancangan sistem *e-commerce*.

a. Model

Courier merupakan model yang memiliki fungsi untuk mengelola seluruh proses manipulasi data yang terkait dengan data *courier* atau kurir yang terdaftar pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Couriers extends Model  {  use HasFactory;  protected $table = 'couriers';  protected $guarded = [];  } |

Kode Program 4. 14 *Source Code Courier*

Kode program 4. 14 merupakan kode program dari model Courier yaitu model yang memiliki fungsi untuk mengelola seluruh proses manipulasi data yang terkait dengan data *courier* atau kurir yang terdaftar pada sistem *e-commerce*.

b. View

*View* admin.courier.home merupakan *view* pada hak akses *admin* yang merupakan bagian *view* untuk menampilkan halaman berisi data setiap kurir yang terdaftar pada sistem *e-commerce*. Halaman ini juga berisi manajemen penuh CRUD bagi *admin* untuk mengelola data kurir yang terdaftar pada sistem *e- commerce.*

|  |
| --- |
| @extends('/admin/layouts/master',['title'=>'Couriers'])  @section('content')  <section role="main" class="content-body">  <header class="page-header">  <h2>Admin</h2>  <div class="right-wrapper text-right">  <ol class="breadcrumbs pr-3">  <li>  <a href="/admin">  <i class="fas fa-home"></i>  </a>  </li>  <li><span>Couriers</span></li>  </ol>    </div>  </header>  <!-- start: page -->  <div class="row">  <div class="col">  <section class="card">  @include('admin/layouts/notif')  <header class="card-header">  <div class="card-actions">  <a href="#" class="card-action card-action-toggle" data-card-toggle></a>  <a href="#" class="card-action card-action-dismiss" data-card-dismiss></a>  </div>  <a href="#modalForm" class=" btn modal-with-form btn-sm btn-success"><i class="fa fa-plus"></i>  Tambah Kurir</a>  </header>  <div class="card-body">  <table class="dataTable table table-bordered table-striped mb-0 responsive" id="datatable-default">  <thead class="table-dark">  <tr>  <th>No</th>  <th>Nama Kurir</th>  <th class="text-center">Action</th>  </tr>  </thead>  <tbody>  @php  $no = 1;  @endphp  @foreach ($couriers as $kurir)  <tr>  <td>{{$no++}}</td>  <td>{{$kurir->courier}}</td>  <td class="text-center"> <a href="#DeletemodalForm" data-id="{{$kurir->id}}"  class="trash modal-with-form btn btn-sm btn-danger"><i  class="fa fa-trash"></i></a>  <a href="#modalEditForm" data-id="{{$kurir->id}}" data-kurir="{{$kurir->courier}}"  class="edit modal-with-form btn btn-sm btn-primary"><i  class="fas fa-pencil-alt"></i></a></td>  </tr>  @endforeach  </tbody>  </table>  </div>  </section>  </div>  </div>  </section>  <!-- end: page -->  </section>  <div id="modalForm" class="modal-block modal-block-primary mfp-hide">  <section class="card">  <header class="card-header">  <h2 class="card-title">Courier Form</h2>  </header>  <div class="card-body">  <form class="form-horizontal form-bordered form-bordered container" enctype="multipart/form-data"  action="{{Route('couriers-store')}}" method="POST">  @csrf  <div class="form-group row">  <div class="col-lg-12">  <label for="tags-input" class="control-label">nama Kurir</label>  <input type="text" name="courier" class="form-control" required placeholder="Input Nama Kurir">  </div>  </div>  </div>  <footer class="card-footer">  <div class="row">  <div class="col-md-12 text-right">  <button class="btn btn-primary" type="submit">Submit</button>  <button class="btn btn-default modal-dismiss">Cancel</button>  </div>  </div>  </footer>  </form>  </section>  </div>  <div id="modalEditForm" class="modal-block modal-block-primary mfp-hide">  <section class="card">  <header class="card-header">  <h2 class="card-title">Edit Courier Form</h2>  </header>  <div class="card-body">  <form class="form-horizontal form-bordered form-bordered container" id="EditFormDiscounts"  enctype="multipart/form-data" action="" method="POST">  @method('put')  @csrf  <div class="form-group row">  <div class="col-lg-12">  <label for="tags-input" class="control-label">Nama Kurir</label>  <input type="text" id="kurir" value="" name="courier" class="form-control" required  placeholder="Input Nama Kurir">  </div>  </div>  </div>  <footer class="card-footer">  <div class="row">  <div class="col-md-12 text-right">  <button class="btn btn-primary" type="submit">Submit</button>  <button class="btn btn-default modal-dismiss">Cancel</button>  </div>  </div>  </footer>  </form>  </section>  </div>  <div class="modal-block modal-block-primary mfp-hide" id="DeletemodalForm" tabindex="-1" role="dialog"  aria-labelledby="myModalLabel" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <h3 id="myModalLabel">Delete Confirmation</h3>  </div>  <div class="modal-body">  <p class="error-text">  <i class="fa fa-warning modal-icon"></i>  apakah anda yakin ingin menghapus item ini?  </p>  </div>  <div class="modal-footer">  <form id="formDelete" action="#" method="post">  @method('delete')  @csrf  <button class="btn btn-default modal-dismiss" data-dismiss="modal"  aria-hidden="true">Cancel</button>  <button type="submit" class="btn btn-danger">Hapus</button>  </form>  </div>  </div>  </div>  </div>  @endsection  @section('footer')  <script>  $(".dataTable").on('click','.trash', function () {  var id = $(this).data('id');  $('#formDelete').attr('action', '/admin/couriers/destroy/' + id);  });  $(".dataTable").on('click','.edit', function () {  var id = $(this).data('id');  var kurir = $(this).data('kurir');  $('#EditFormDiscounts').attr('action', '/admin/couriers/update/' + id);  $('#kurir').attr('value', kurir);    });  </script>  @endsection |

Kode Program 4. 15 *Source Code* *View Courier*

Kode program 4. 15 merupakan kode pogram dari v*iew* admin.courier.home yaitu *view* pada hak akses *admin* yang merupakan bagian *veiw* untuk menampilkan halaman berisi data setiap kurir yang terdaftar pada sistem *e-commerce*. Halaman ini juga berisi manajemen penuh CRUD bagi *admin* untuk mengelola data kurir yang terdaftar pada sistem *e-commerce*.

c. Controller

*Controller* CourierController merupakan *controller* yang menghubungkan model Courier dengan bagian *view* admin.courier.home dan admin.courier.edit pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Http\Controllers\Admin;  use App\Models\Couriers;  use Illuminate\Http\Request;  use App\Http\Controllers\Controller;  class CouriersController extends Controller  {  /\*\*  \* Display a listing of the resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function index()  {  $couriers = Couriers::all();  return view('admin/couriers/index',compact('couriers'));  }  /\*\*  \* Show the form for creating a new resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function create()  {  //  }  /\*\*  \* Store a newly created resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @return \Illuminate\Http\Response  \*/  public function store(Request $request)  {  $couriers = $request->all();  $couriers['code'] = strtolower($request->courier);  Couriers::create($couriers);  return redirect()->back()->with('success','Berhasil Menambahkan Kurir');  }  /\*\*  \* Display the specified resource.  \*  \* @param \App\Models\Couriers $couriers  \* @return \Illuminate\Http\Response  \*/  public function show(Couriers $couriers)  {  //  }  /\*\*  \* Show the form for editing the specified resource.  \*  \* @param \App\Models\Couriers $couriers  \* @return \Illuminate\Http\Response  \*/  public function edit(Couriers $couriers)  {  //  }  /\*\*  \* Update the specified resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @param \App\Models\Couriers $couriers  \* @return \Illuminate\Http\Response  \*/  public function update(Request $request, Couriers $couriers)  {  $courier = $request->all();  $courier['code'] = strtolower($request->courier);  $couriers->update($courier);  return redirect()->back()->with('info','Berhasil Mengupdate Kurir');  }  /\*\*  \* Remove the specified resource from storage.  \*  \* @param \App\Models\Couriers $couriers  \* @return \Illuminate\Http\Response  \*/  public function destroy(Couriers $couriers)  {  $couriers->delete();  return redirect()->back()->with('error','Data Telah Dihapus');    }  } |

Kode Program 4. 16 *Source Code* *Courier Controller*

Kode program 4. 16 merupakan kode program dari *controller* CourierController yaitu *controller* yang menghubungkan model Courier dan bagian *view* admin.courier.home dan admin.courier.edit pada sistem *e- commerce*.

#### 4.2.2.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi CRUD pada tabel *courier* dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari CRUD tabel *courier* pada rancangan *e- commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 15 Halaman CRUD Tabel *Courier*

Gambar 4. 15 merupakan halaman CRUD untuk tabel *courier* yang memuat daftar kurir yang telah terdaftar pada sebuah *e-commerce*. *Admin* dapat mendaftarkan lebih banyak data kurir*,* mengubah detail kurir, melihat detail kurir, dan menghapus kurir yang telah terdaftar.

|  |
| --- |
|  |

Gambar 4. 16 Halaman *Create* pada Tabel *Courier*

Gambar 4. 16 merupakan halaman dimana *admin* dapat menambahkan atau mendaftarkan kurir baru pada *e-commerce* (*create*). *Admin* diminta memasukkan nama kurir yang akan didaftarkan.

|  |
| --- |
|  |

Gambar 4. 17 Halaman CRUD Tabel *Courier*

Gambar 4.17 menunjukkan halaman CRUD *courier* dimana pada halaman tersebut ditunjukkan bahwa kurir yang tadi didaftarkan oleh *admin* sudah berhasil tersimpan atau terdaftar pada *e-commerce.*

|  |
| --- |
|  |

Gambar 4. 18 Halaman Detail *Courier*

Gambar 4. 18 merupakan halaman mengubah detail kurir dimana *admin* dapat mengubah detail (*update*) dari kurir yang telah terdaftar pada sebuah sistem *e-commerce.*

|  |
| --- |
|  |

Gambar 4. 19 Halaman Detail Data Kurir

Gambar 4. 19 menunjukkan halaman CRUD *courier* dimana pada halaman tersebut ditunjukkan bahwa detail data kurir pada sistem *e-commerce* telah berhasil diubah oleh *admin*.

|  |
| --- |
|  |

Gambar 4. 20 Tampilan *Admin* Berhasil Menghapus Data pada Tabel *Courier*

Gambar 4. 20 menunjukkan bahwa *admin* berhasil menghapus data kurir yang baru saja didaftarkan dengan cara menekan tombol *delete* pada halaman CRUD di sebuah *e-commerce*.

### 4.2.3 CRUD pada Tabel Category

Berikut ini akan ditampilkan kode program CRUD (*create, read, update, delete*) pada tabel *category* pada sistem *e-commerce* dengen menerapkan konsep MVC.

#### 4.2.3.1 Source Code

CRUD (*create, read, update, delete*) pada tabel *category* diterapkan dengan menggunakan konsep MVC. Berikut ini merupakan kode program dalam implementasi CRUD pada tabel *category* dalam rancangan sistem *e-commerce*.

a. Model

Category merupakan model yang memiliki fungsi untuk mengelola seluruh proses manipulasi data yang terkait dengan data *category* yang terdaftar pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Category extends Model  {  use HasFactory;  protected $table = 'product\_categories';  protected $guarded = [];  public function product\_category\_detail(){  return $this->hasMany(ProductCategoryDetails::class,'category\_id','id');  }  public function product(){  return $this->belongsToMany(Products::class,'product\_category\_details','category\_id', 'product\_id')->withPivot('id');  }  } |

Kode Program 4. 17 *Source Code Category*

Kode program 4. 17 merupakan kode program dari model Category yaitu model yang memiliki fungsi untuk mengelola seluruh proses manipulasi data yang terkait dengan data *category* yang terdaftar pada sistem *e-commerce.*

b. View

*View* admin.category.home merupakan *view* pada hak akses *admin* yang merupakan bagian *view* untuk menampilkan halaman berisi data setiap kategori yang terdaftar pada sistem *e-commerce*. Halaman ini juga berisi manajemen penuh CRUD bagi *admin* untuk mengelola data kategori yang terdaftar pada sistem *e- commerce.*

|  |
| --- |
| @extends('/layouts/master',['title'=>'Dashboard'])  @section('content')  <section role="main" class="content-body">  <header class="page-header">  <h2>Admin</h2>    <div class="right-wrapper text-right">  <ol class="breadcrumbs pr-3">  <li>  <a href="/admin">  <i class="fas fa-home"></i>  </a>  </li>  <li>  <a href="{{Route('category')}}">  <span>Category</span>  </a>  </li>  <li><span>Tambah Category</span></li>    </ol>      </div>  </header>  <!-- start: page -->  <div class="row">  <div class="col">  <section class="card">  <header class="card-header">  <div class="card-actions">  <a href="#" class="card-action card-action-toggle" data-card-toggle></a>  <a href="#" class="card-action card-action-dismiss" data-card-dismiss></a>  </div>    <h4 class="card-title">Add Form</h4>  </header>  <div class="card-body">  <form class="form-horizontal form-bordered form-bordered container" enctype="multipart/form-data" action="{{Route('store-category')}}" method="POST">  @csrf  <div class="form-group row">  <div class="col-lg-12">  <label for="tags-input" class="control-label">Input Category</label>  {{-- <input type="text" name="categorys[]" multiple id="tags-input" data-role="tagsinput" data-tag-class="badge badge-primary" class="form-control" />  --}}  <select name="categorys[]" id="" multiple id="tags-input" data-role="tagsinput" data-tag-class="badge badge-primary" class="form-control">    </select>  <small class="text-danger">\* Silahkan masukkan nama category (dapat memasukan lebih dari satu nama)</small>  </div>  </div>  <button class="float-right btn btn-sm btn-success" type="submit">Submit</button>  </form>  </div>  </section>  </div></div>  </section>  @endsection  @section('footer')  <!-- Specific Page Vendor -->  <script src="/assets/vendor/jqueryui-touch-punch/jquery.ui.touch-punch.js"></script>  <script src="/assets/vendor/select2/js/select2.js"></script>  <script src="/assets/vendor/jquery-maskedinput/jquery.maskedinput.js"></script>  <script src="/assets/vendor/bootstrap-tagsinput/bootstrap-tagsinput.js"></script>  <!-- Examples -->  <script src="/assets/js/examples/examples.advanced.form.js"></script>  @endsection |

Kode Program 4. 18 *Source Code View Admin Category*

Kode program 4. 18 merupakan kode program dari *view* admin.category.home yaitu bagian *view* untuk menampilkan halaman berisi data setiap kategori yang terdaftar pada sistem *e-commerce*. Halaman ini juga berisi manajemen penuh CRUD bagi admin untuk mengelola data kategori yang terdaftar pada sistem *e-commerce*.

c. Controller

*Controller* CategoryController merupakan *controller* yang menghubungkan model Category dan bagian *view* admin.category.home dan admin.category.edit pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Http\Controllers\Admin;  use App\Models\Category;  use Illuminate\Http\Request;  use App\Http\Controllers\Controller;  class CategoryController extends Controller  {  /\*\*  \* Display a listing of the resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function index()  {  $category = Category::all();  return view('/admin/category/index', compact('category'));  }  /\*\*  \* Show the form for creating a new resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function create()  {  return view('/admin/category/add');  }  /\*\*  \* Store a newly created resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @return \Illuminate\Http\Response  \*/  public function store(Request $request)  {  foreach ($request->categorys as $key => $value) {  Category::create([  'category\_name' => $request->categorys[$key]  ]);  }  return redirect()->back()->with('success', 'berhasil menambahkan category');  }  /\*\*  \* Display the specified resource.  \*  \* @param \App\Models\Category $category  \* @return \Illuminate\Http\Response  \*/  public function show(Category $category)  {  //  }  /\*\*  \* Show the form for editing the specified resource.  \*  \* @param \App\Models\Category $category  \* @return \Illuminate\Http\Response  \*/  public function edit(Category $category)  {  //  }  /\*\*  \* Update the specified resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @param \App\Models\Category $category  \* @return \Illuminate\Http\Response  \*/  public function update(Request $request, Category $category)  {  $category->category\_name = $request->category\_name;  $category->update();  return redirect()->back()->with('info','update category berhasil');  }  /\*\*  \* Remove the specified resource from storage.  \*  \* @param \App\Models\Category $category  \* @return \Illuminate\Http\Response  \*/  public function destroy(Category $category)  {  $category->delete();  return redirect()->back()->with('error','categori telah Terhapus');  }  } |

Kode Program 4. 19 *Source Code Category Controller*

Kode program 4. 19 merupakan kode program dari *controller* CategoryController yaitu *controller* yang menghubungkan model Category dan bagian *view* admin.category.home dan admin.category.edit pada sistem *e- commerce*.

#### 4.2.3.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi CRUD pada tabel *category* dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari CRUD tabel *category* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 21 Tampilan Halaman CRUD Tabel *Category*

Gambar 4. 21 merupakan halaman CRUD untuk tabel *category* yang memuat daftar kategori yang telah terdaftar pada sebuah *e-commerce*. *Admin* dapat mendaftarkan lebih banyak data kategori*,* mengubah detail kategori, melihat detail kategori, dan menghapus kategori yang telah terdaftar.

|  |
| --- |
|  |

Gambar 4. 22 Tampilan Halaman *Create* *Product*

Gambar 4. 22 merupakan halaman dimana *admin* dapat menambahkan atau mendaftarkan kategori produk baru pada *e-commerce* (*create*). *Admin* diminta memasukkan nama kategori yang akan didaftarkan.

|  |
| --- |
|  |

Gambar 4. 23 Tampilan Halaman CRUD *Category*

Gambar 4. 23 menunjukkan halaman CRUD *category* dimana pada halaman tersebut ditunjukkan bahwa kategori yang tadi didaftarkan oleh admin sudah berhasil tersimpan atau terdaftar pada *e-commerce*.

|  |
| --- |
|  |

Gambar 4. 24 Tampilan Halaman Ubah Detail Kategori

Gambar 4. 24 merupakan halaman mengubah detail kategori dimana *admin* dapat mengubah detail (*update*) dari kategori yang telah terdaftar pada sebuah sistem *e-commerce.*

|  |
| --- |
|  |

Gambar 4. 25 Tampilan Halaman Detail Kategori yang Berhasil Diubah

Gambar 4. 25 menunjukkan halaman CRUD *category* dimana pada halaman tersebut ditunjukkan bahwa detail data kategori telah berhasil diubah oleh *admin.*

|  |
| --- |
|  |

Gambar 4. 26 Tampilan Halaman Hapus Kategori

Gambar 4. 26 menunjukkan bahwa *admin* berhasil menghapus data kategori yang baru saja didaftarkan dengan cara menekan tombol *delete* pada halaman CRUD di sebuah *e-commerce*.

## 4.3 Transaksi

Modul ketiga praktikum pemrograman internet yaitu pengimplementasian proses transaksi pada rancangan sistem *e-commerce* yaitu proses pembelian produk dan ulasan produk. Dimana saat proses pembelian produk admin dapat mengetahui dan mengganti status transaksi, sementara pada proses ulasan produk, pembeli dapat memberikan ulasan dengan nilai 1 sampai dengan 5 mengenai produk yang telah dibeli. Untuk mendapatkan ongkos kirim untuk masing-masing transaksi pembelian, digunakan API seperti RajaOngkir API.

### 4.3.1 Add to Chart

Proses transaksi pertama yang akan diterapkan yaitu *add to cart* dimana *user* dapat melakukan transaksi pembelian produk langsung maupun menambahkan ke *cart* untuk jenis barang lebih dari satu. Berikut ini merupakan *source code* dan tampilan dari transaksi *add to cart.*

#### 4.3.1.1 Source Code

*Source code* dari transaksi pembelian produk dan *add to cart* dimana pengguna dapat menambahkan barang pada *cart* atau melakukan pembelian produk langsung diterapkan dengan menggunakan metode MVC sebagai berikut.

a. Model

Model untuk untuk proses *add to cart* yaitu model Cart. Berikut merupakan kode program untuk model Cart untuk proses *add to cart* pada sistem *e-commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Carts extends Model  {  use HasFactory;  protected $table = 'carts';  protected $guarded = [];  public function products(){  return $this->belongsTo(Products::class, 'product\_id', 'id');  }  public function user(){  return $this->belongsTo(User::class, 'user\_id', 'id');  }  } |

Kode Program 4. 20 *Source Code Cart*

Kode program 4. 20 merupakan kode program untuk model Cart dimana mendefinisikan proses *add to cart* yang dilakukan oleh pengguna untuk transaksi pembelian beberapa barang.

c. View

Terdapat sejumlah *view* yang digunakan dalam sebuah proses transaksi pembelian langsung maupun transaksi pembelian melalui *cart*, di antaranya *view* mycart untuk melihat *list* barang yang telah dimasukan ke dalam *cart*, cart\_courier\_service untuk memilih kurir dari transaksi pembelian, cart\_payment untuk menampilkan halaman memilih *billing address* dan untuk transaksi pembelian langsung dengan *view* product\_courier\_service untuk memilih jenis kurir dan product\_payment untuk memilih *billing address*. Berikut ini adalah *view* yang digunakan dalam proses transaksi pembelian langsung dan *add to cart.*

|  |
| --- |
| @extends('layouts/master')  @section('content')  <div class="banner-wrapper has\_background">  <div class="banner-wrapper-inner">  <div role="navigation" aria-label="Breadcrumbs" class="breadcrumb-trail breadcrumbs container">  <ul class="trail-items breadcrumb">  <li class="trail-item trail-begin"><a href="/"><span>Home</span></a></li>  <li class="trail-item trail-end active"><span>Cart</span>  </li>  </ul>  </div>  </div>  </div>  <main class="site-main main-container no-sidebar">  <div class="container">  <div class="row">  <div class="main-content col-md-12">  <div class="page-main-content">  <div class="BLitzz">  <div class="BLitzz-notices-wrapper"></div>  <div class="BLitzz-cart-form">  <table class="shop\_table shop\_table\_responsive cart BLitzz-cart-form\_\_contents"  cellspacing="0">  <thead>  <tr>  <th class="product-remove">&nbsp;</th>  <th class="product-name">Product</th>  <th class="product-price">Price</th>  <th class="product-quantity">Quantity</th>  <th class="product-quantity">Total</th>  </tr>  </thead>  <tbody>  @foreach ($carts as $cart)  <tr class="BLitzz-cart-form\_\_cart-item cart\_item">  <td class="product-remove">  <form action="{{Route('delete',['cart'=>$cart->id])}}" method="post">  @method('delete')  @csrf  <button type="submit" class="btn btn-danger"  aria-label="Remove this item" data-product\_id="27"  data-product\_sku="885B712"><i class="fa fa-trash"></i></button>  </form>  </td>    <td class="product-name" data-title="Product">  <a href="#">{{$cart->products->product\_name}}</a></td>  <td class="product-price" data-title="Price">  <span class="BLitzz-Price-amount amount">  @php  $is\_discount = false;  @endphp  @foreach ($cart->products->discounts as  $discount)  @if (date('Y-m-d') >= $discount->start  && date('Y-m-d') < $discount->end)  @php  $diskon = ($discount->percentage /  100) \* $cart->products->price;  @endphp  @if ($diskon)  @php  $is\_discount = true;  @endphp  {{"Rp " . number\_format($cart->products->price - $diskon,2,',','.')}}  <br>  @endif  @endif  @endforeach  @if ($is\_discount)  <small><strike>{{ "Rp " . number\_format($cart->products->price, 2, ',', '.')}}</strike></small>  @else  {{"Rp " . number\_format($cart->products->price,2,',','.')}}  @endif  </span></td>  <td class="product-quantity" data-title="Quantity">  <div class="quantity">  <span class="qty-label">Quantiy:</span>  <div class="control">  <div>  <input type="hidden" name="id[]" value="{{$cart->id}}">  <div data-min="1"  class="dec button btn-number qtyplus quantity-minus">-  </div>  <input type="text" class="incdec input-qty input-text"  name="qty[]" id="value{{$cart->id}}" value="{{$cart->qty}}"  data-price="@if($is\_discount){{$cart->products->price - $diskon}}@else{{$cart->products->price}}@endif" />  <div class="inc button btn-number qtyplus quantity-plus"  data-max="{{$cart->products->stock}}">+</div>  </div>  </div>  </div>  </td>  <td class="product-price" data-title="Price">  <span class="BLitzz-Price-amount amount subtotal">  @if($is\_discount)  {{"Rp " . number\_format(($cart->products->price-$diskon)\*$cart->qty,0,',','.')}}@else  {{"Rp " . number\_format($cart->products->price\*$cart->qty,0,',','.')}}@endif  </span></td>  </tr>  @endforeach  <tr>  <td colspan="6" class="actions">  <button type="button" id="submit" class="button" name="update\_cart"  value="Update cart">Update cart  </button>  </td>  </tr>  </tbody>  </table>  </div>  <div class="cart-collaterals">  <div class="cart\_totals ">  <h2>Cart totals</h2>  <table class="shop\_table shop\_table\_responsive" cellspacing="0">  <tbody>  <tr class="order-total">  <th>Total</th>  <td data-title="Total">  <strong><span class="BLitzz-Price-amount amount sub-total">  <span  class="BLitzz-Price-currencySymbol-totals">{{"Rp " . number\_format($total,2,',','.')}}</span>  </span></strong>  </td>  </tr>  </tbody>  </table>  <div class="BLitzz-proceed-to-checkout @if ($carts->count() < 1) d-none @endif">  <a href="{{Route('checkout')}}" class="checkout-button button alt BLitzz-forward">  Proceed to checkout</a>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </main>  @endsection  @section('scripts')  <script>  $(document).on('click','#submit',function(){  var qty = $('input[name="qty[]"]').map(function(){  return this.value;  }).get();  var id = $('input[name="id[]"]').map(function(){  return this.value;  }).get();  $.ajax({  url: '{{Route("update-cart")}}',  type: 'post',  data: {  'qty[]': qty,  'id[]': id,  // other data  },  success: function(data){  loadcount();  loadcarts();  loadtotal();  if (data == 'stok habis') {  toastr.error(data);  }else{  toastr.info(data);  }    }  });  });  </script>  @endsection |

Kode Program 4. 21 *Source Code View MyCart*

Kode program 4. 21 merupakan kode program untuk *view* mycart dimana *view* ini mengatur tampilan dari halaman *cart* dari seorang pengguna berisi *list* produk yang ingin di *checkout.*

|  |
| --- |
| @extends('/admin/layouts/master',['title'=>'Couriers'])  @section('content')  <section role="main" class="content-body">  <header class="page-header">  <h2>Admin</h2>  <div class="right-wrapper text-right">  <ol class="breadcrumbs pr-3">  <li>  <a href="/admin">  <i class="fas fa-home"></i>  </a>  </li>  <li><span>Couriers</span></li>  </ol>    </div>  </header>  <!-- start: page -->  <div class="row">  <div class="col">  <section class="card">  @include('admin/layouts/notif')  <header class="card-header">  <div class="card-actions">  <a href="#" class="card-action card-action-toggle" data-card-toggle></a>  <a href="#" class="card-action card-action-dismiss" data-card-dismiss></a>  </div>  <a href="#modalForm" class=" btn modal-with-form btn-sm btn-success"><i class="fa fa-plus"></i>  Tambah Kurir</a>  </header>  <div class="card-body">  <table class="dataTable table table-bordered table-striped mb-0 responsive" id="datatable-default">  <thead class="table-dark">  <tr>  <th>No</th>  <th>Nama Kurir</th>  <th class="text-center">Action</th>  </tr>  </thead>  <tbody>  @php  $no = 1;  @endphp  @foreach ($couriers as $kurir)  <tr>  <td>{{$no++}}</td>  <td>{{$kurir->courier}}</td>  <td class="text-center"> <a href="#DeletemodalForm" data-id="{{$kurir->id}}"  class="trash modal-with-form btn btn-sm btn-danger"><i  class="fa fa-trash"></i></a>  <a href="#modalEditForm" data-id="{{$kurir->id}}" data-kurir="{{$kurir->courier}}"  class="edit modal-with-form btn btn-sm btn-primary"><i  class="fas fa-pencil-alt"></i></a></td>  </tr>  @endforeach  </tbody>  </table>  </div>  </section>  </div>  </div>  </section>  <!-- end: page -->  </section>  <div id="modalForm" class="modal-block modal-block-primary mfp-hide">  <section class="card">  <header class="card-header">  <h2 class="card-title">Courier Form</h2>  </header>  <div class="card-body">  <form class="form-horizontal form-bordered form-bordered container" enctype="multipart/form-data"  action="{{Route('couriers-store')}}" method="POST">  @csrf  <div class="form-group row">  <div class="col-lg-12">  <label for="tags-input" class="control-label">nama Kurir</label>  <input type="text" name="courier" class="form-control" required placeholder="Input Nama Kurir">  </div>  </div>  </div>  <footer class="card-footer">  <div class="row">  <div class="col-md-12 text-right">  <button class="btn btn-primary" type="submit">Submit</button>  <button class="btn btn-default modal-dismiss">Cancel</button>  </div>  </div>  </footer>  </form>  </section>  </div>  <div id="modalEditForm" class="modal-block modal-block-primary mfp-hide">  <section class="card">  <header class="card-header">  <h2 class="card-title">Edit Courier Form</h2>  </header>  <div class="card-body">  <form class="form-horizontal form-bordered form-bordered container" id="EditFormDiscounts"  enctype="multipart/form-data" action="" method="POST">  @method('put')  @csrf  <div class="form-group row">  <div class="col-lg-12">  <label for="tags-input" class="control-label">Nama Kurir</label>  <input type="text" id="kurir" value="" name="courier" class="form-control" required  placeholder="Input Nama Kurir">  </div>  </div>  </div>  <footer class="card-footer">  <div class="row">  <div class="col-md-12 text-right">  <button class="btn btn-primary" type="submit">Submit</button>  <button class="btn btn-default modal-dismiss">Cancel</button>  </div>  </div>  </footer>  </form>  </section>  </div>  <div class="modal-block modal-block-primary mfp-hide" id="DeletemodalForm" tabindex="-1" role="dialog"  aria-labelledby="myModalLabel" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <h3 id="myModalLabel">Delete Confirmation</h3>  </div>  <div class="modal-body">  <p class="error-text">  <i class="fa fa-warning modal-icon"></i>  apakah anda yakin ingin menghapus item ini?  </p>  </div>  <div class="modal-footer">  <form id="formDelete" action="#" method="post">  @method('delete')  @csrf  <button class="btn btn-default modal-dismiss" data-dismiss="modal"  aria-hidden="true">Cancel</button>  <button type="submit" class="btn btn-danger">Hapus</button>  </form>  </div>  </div>  </div>  </div>  @endsection  @section('footer')  <script>  $(".dataTable").on('click','.trash', function () {  var id = $(this).data('id');  $('#formDelete').attr('action', '/admin/couriers/destroy/' + id);  });  $(".dataTable").on('click','.edit', function () {  var id = $(this).data('id');  var kurir = $(this).data('kurir');  $('#EditFormDiscounts').attr('action', '/admin/couriers/update/' + id);  $('#kurir').attr('value', kurir);    });  </script>  @endsection |

Kode Program 4. 22 *Source Code View Cart Courier Service*

Kode program 4. 22 merupakan kode program untuk *view* cart\_courier\_service dimana *view* ini mengatur tampilan dari halaman pengguna dapat memilih jasa kurir yang diinginkan untuk pengiriman ke alamat pengguna.

|  |
| --- |
| @extends('layouts/master')  @section('content')  <div class="banner-wrapper has\_background">  <img src="assets/images/slide35.png" class="img-responsive attachment-1920x447 size-1920x447" alt="img">  <div class="banner-wrapper-inner">  <h1 class="page-title container">Transaction</h1>  <div role="navigation" aria-label="Breadcrumbs" class="breadcrumb-trail breadcrumbs container">  <ul class="trail-items breadcrumb">  <li class="trail-item trail-begin"><a href="/"><span>Home</span></a></li>  <li class="trail-item trail-end active"><span>Transaction</span>  </li>  </ul>  </div>  </div>  </div>  <main class="site-main main-container no-sidebar">  <div class="container">  <div class="row">  <div class="main-content col-md-12">  <div class="page-main-content">  <div class="BLitzz">  <div class="BLitzz-notices-wrapper"></div>  <form class="BLitzz-cart-form">  <table class="shop\_table shop\_table\_responsive cart BLitzz-cart-form\_\_contents"  cellspacing="0">  <thead>  <tr>  <th class="product-no">No</th>  <th class="product-adress">Alamat</th>  <th class="product-telp">Telepon</th>  <th class="product-status">Status</th>  <th class="product-subtotal">Total</th>  <th class="product-action">action</th>  </tr>  </thead>  <tbody>  @php  $no = 1;  @endphp  @foreach ($transaksis ->reverse() as $transaksi)  <tr class="BLitzz-cart-form\_\_cart-item cart\_item">  <td class="product-remove">  {{$no++}}</td>  <td class="address">  <p>{{$transaksi->address}}</p></td>  <td class="status">  <p>{{$transaksi->telp}}</p></td>  <td class="telp">  <p>{{$transaksi->status}}</p>  </td>  <td class="product-subtotal">  <span class="BLitzz-Price-amount amount">{{"Rp " . number\_format($transaksi->total,2,',','.')}}</span></td>  <td class="action text-center">  <a href="{{Route('detail-transaksi',['transactions' => encrypt($transaksi->id) ])}}">Detail</a></td>  </tr>  @endforeach  </tbody>  </table>  </form>  </div>  </div>  </div>  </div>  </div>  </main>  @endsection |

Kode Program 4. 23 *Source Code View Cart Payment*

Kode program 4. 23 merupakan kode program untuk *view* cart\_payment dimana *view* ini mengatur tampilan dari halaman pengguna dapat memilih alamat *billing.*

|  |
| --- |
| @extends('layouts/master')  @section('content')  <div class="banner-wrapper has\_background">  <img src="assets/images/slide35.png" class="img-responsive attachment-1920x447 size-1920x447" alt="img">  <div class="banner-wrapper-inner">  <h1 class="page-title container">Checkout</h1>  <div role="navigation" aria-label="Breadcrumbs" class="breadcrumb-trail breadcrumbs container">  <ul class="trail-items breadcrumb">  <li class="trail-item trail-begin"><a href="/"><span>Home</span></a></li>  <li class="trail-item trail-end active"><span>Checkout</span>  </li>  </ul>  </div>  </div>  </div>  @if ($errors->has('total'))  <h3 class="text-danger text-center">{{ $errors->first('total') }}</h3>  @elseif($errors->has('sub\_total'))  <h3 class="text-danger text-center">{{ $errors->first('sub\_total') }}</h3>  @endif  <main class="site-main main-container no-sidebar">  <div class="container">  <div class="row">  <div class="main-content col-md-12">  <div class="page-main-content">  <div class="BLitzz">  <div class="BLitzz-notices-wrapper"></div>  <form action="{{Route('checkout-all')}}" method="post" class="checkout BLitzz-checkout"  enctype="multipart/form-data" novalidate="novalidate">  @csrf  <div class="col2-set" id="customer\_details">  <div class="col-1">  <div class="BLitzz-billing-fields">  <h3>Billing details</h3>  <div class="BLitzz-billing-fields\_\_field-wrapper">  <p class="form-row form-row-wide addresses-field update\_totals\_on\_change validate-required"  id="billing\_province\_field" data-priority="40"><label  for="billing\_province" class="">Province&nbsp;<abbr class="required"  title="required">\*</abbr></label>  <span class="BLitzz-input-wrapper">  <select id="provinsi" name="province" required  class="province\_to\_state province\_select"  autocomplete="province" tabindex="-1" aria-hidden="true">  <option value="" selected disabled>Pilih Provinsi</option>  @foreach ($provinsis as $provinsi)  <option data-id="{{$provinsi->id}}"  value="{{$provinsi->title}}">{{$provinsi->title}}</option>  @endforeach  </select>  </span>  @if ($errors->has('province'))  <span class="text-danger">{{ $errors->first('province') }}</span>  @endif  </p>  <p class="form-row form-row-wide addresses-field update\_totals\_on\_change validate-required"  id="billing\_regency\_field" data-priority="40"><label  for="billing\_regency" class="">Kota&nbsp;<abbr class="required"  title="required">\*</abbr></label>  <span class="BLitzz-input-wrapper">  <select id="kota" name="regency" disabled required  class="regency\_to\_state regency\_select" autocomplete="regency"  tabindex="-1" aria-hidden="true">  <option value="" selected disabled>Pilih Kota</option>  </select>  </span>  @if ($errors->has('regency'))  <span class="text-danger">{{ $errors->first('regency') }}</span>  @endif  </p>  <p class="form-row form-row-wide alamat-field validate-required"  id="billing\_alamat\_1\_field" data-priority="50"><label  for="billing\_alamat\_1" class="">Alamat&nbsp;<abbr class="required"  title="required">\*</abbr></label><span  class="BLitzz-input-wrapper"><input required type="text"  class="input-text " name="address" id="billing\_alamat\_1"  placeholder="Alamat Lengkap" value=" {{old('address')}}"  autocomplete="alamat-line1"  data-placeholder="Alamat Lengkap"></span>  @if ($errors->has('address'))  <span class="text-danger">{{ $errors->first('address') }}</span>  @endif  </p>  <p class="form-row form-row-wide alamat-field validate-required"  id="billing\_telp\_field" data-priority="50"><label for="billing\_tepl"  class="">Nomor Telepon&nbsp;<abbr class="required"  title="required">\*</abbr></label><span  class="BLitzz-input-wrapper"><input required type="text"  class="input-text " name="telp" id="billing\_tepl"  placeholder="Nomor Telepon" value="{{old('telp')}}" autocomplete="alamat-line1"  data-placeholder="Nomor Telepon"></span>  @if ($message = Session::get('false'))  <span class="text-danger">{{ $message }}</span>  @elseif($errors->has('telp'))  <span class="text-danger">{{ $errors->first('telp') }}</span>  @endif  </p>  <p class="form-row form-row-wide addresses-field update\_totals\_on\_change validate-required"  id="billing\_courier\_field" data-priority="40"><label  for="billing\_courier" class="">Kurir&nbsp;<abbr class="required"  title="required">\*</abbr></label>  <span class="BLitzz-input-wrapper">  <select name="courier\_id" id="kurir" required>  <option value="" selected disabled>Pilih Kurir</option>  @foreach ($kurirs as $kurir)  <option data-kurir="{{$kurir->code}}" value="{{$kurir->id}}">  {{$kurir->courier}}</option>  @endforeach  </select>  </span>  @if ($errors->has('courier\_id'))  <span class="text-danger">{{ $errors->first('courier\_id') }}</span>  @endif  </p>  <p class="form-row form-row-wide addresses-field update\_totals\_on\_change validate-required"  id="billing\_courier\_field" data-priority="40"><label  for="billing\_courier" class="">Layanan&nbsp;<abbr class="required"  title="required">\*</abbr></label>  <span class="BLitzz-input-wrapper">  <select name="shipping\_cost" id="layanan" required>  <option value="" selected disabled>Pilih Layanan</option>  </select>  </span>  @if ($errors->has('shipping\_cost'))  <span class="text-danger">{{ $errors->first('shipping\_cost') }}</span>  @endif  </p>  </div>  </div>  </div>  </div>  <h3 id="order\_review\_heading">Your order</h3>  <div id="order\_review" class="BLitzz-checkout-review-order">  <table class="shop\_table BLitzz-checkout-review-order-table">  <thead>  <tr>  <th class="product-name">Product</th>  <th class="product-total">Total</th>  </tr>  </thead>  <tbody>  @foreach ($carts as $cart)  <tr class="cart\_item">  <td class="product-name">  {{$cart->products->product\_name}}&nbsp;&nbsp; <strong  class="product-quantity">×  {{$cart->qty}}</strong></td>  <td class="product-total">  <span class="BLitzz-Price-amount amount">  @php  $is\_discount = false;  @endphp  @foreach ($cart->products->discounts as  $discount)  @if (date('Y-m-d') >= $discount->start  && date('Y-m-d') < $discount->end)  @php  $diskon = ($discount->percentage /  100) \* $cart->products->price;  @endphp  @if ($diskon)  @php  $is\_discount = true;  @endphp  <span id="hargadiskon"  data-harga="{{$cart->products->price - $diskon}}">{{"Rp " . number\_format($cart->products->price - $diskon,2,',','.')}}</span>  <input type="hidden" name="harga\_diskon[]" multiple  value="{{$cart->products->price - $diskon ?? '0'}}">  <input type="hidden" name="diskon[]" multiple  value="{{$discount->percentage}}">  <br>  @endif  @endif  @endforeach  @if ($is\_discount)  <small><strike>{{ "Rp " . number\_format($cart->products->price, 2, ',', '.')}}</strike></small>  @else  <input type="hidden" name="diskon[]" multiple value="0">  {{"Rp " . number\_format($cart->products->price,2,',','.')}}  <input type="hidden" name="harga\_diskon[]" multiple  value="{{$cart->products->price}}">  @endif  </span>  </td>  </tr>  @endforeach  </tbody>  <tfoot>  <tr class="cart-subtotal">  <th>Sub Total</th>  <td><span  class="BLitzz-Price-amount amount">{{"Rp " . number\_format($total,0,',','.')}}</span>  </td>  </tr>  <tr class="cart-subtotal">  <th>Ongkir</th>  <td><span class="BLitzz-Price-amount amount" id="ongkir">0</span></td>  </tr>  <tr class="cart-subtotal">  <th>Berat Total</th>  <td><span class="BLitzz-Price-amount amount" id="berat"  data-berat="{{$berattotal}}">{{$berattotal}} g</span>  </td>  </tr>  <tr class="order-total">  <th>Total</th>  <td><strong><span data-total="{{$total}}" id="total"  class="BLitzz-Price-amount amount">{{"Rp " . number\_format($total,0,',','.')}}</span></strong>  </td>  </tr>  </tfoot>  </table>  <div>  <button type="submit" class="button alt" id="place\_order" value="Place order"  data-value="Place order">Place  order  </button>  <input type="hidden" id="BLitzz-process-checkout-nonce"  value="634590c981"><input type="hidden"  value="/BLitzz/?BLitzz-ajax=update\_order\_review">  </div>  </div>  </form>  </div>  </div>  </div>  </div>  </div>  </main>  @endsection  @section('scripts')  <script>  $.ajaxSetup({  headers: {  'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  }  });  $('#provinsi').on('change', function() {  var id = $(this).find('option:selected').data("id");  var html\_option ='';  $('#kota').prop("disabled", false);  $.ajax({  url: '{{Route("getkota")}}',  type: 'get',  data: {id: id},  dataType: 'json',  success: function(data){  $.each(data.kota, function(i, kotas) {  html\_option += '<option data-kota='+kotas.city\_id+' value='+kotas.title+'>'+kotas.title+'</option>'  });  $('#kota').html(html\_option);  var kurir = $('#kurir :selected').val();  if (kurir) {  loadongkir();  }  }  });  });  $('#kota').on('change', function() {  var kurir = $('#kurir :selected').val();  if (kurir) {  loadongkir();  }    });  $('#kurir').on('change', function() {  loadongkir();  });  $('select[name="shipping\_cost"]').on('change', function() {  loadSubOngkir();  loadtotals();  });  function loadtotals() {  var ongkir = $('select[name="shipping\_cost"] :selected').val();  var total = $('#total').attr('data-total');  var totals = Number(ongkir) + Number(total);  $('span#total').html('Rp '+format(totals));  }  function loadSubOngkir() {  var html\_option = '';  html\_option += $('select[name="shipping\_cost"] :selected').val();  $('#ongkir').html('Rp '+format(html\_option));  }  function loadongkir(){  var kurir =$('#kurir').find('option:selected').data("kurir");  var kota = $('#kota').find('option:selected').data("kota");  var berat = $('#berat').data('berat');  var html\_option = '';  $.ajax({  url: '{{Route("cekongkir")}}',  type: 'post',  data: {  kurir: kurir,  kota: kota,  berat: berat  },  success:function(data){    $('select[name="shipping\_cost"]').html('<option value="" selected>Tidak ada layanan</option>');    // looping data result nya  $.each(data, function(key, value){  // looping data layanan misal jne reg, jne oke, jne yes  $.each(value.costs, function(key1, value1){  // untuk looping cost nya masing masing  $.each(value1.cost, function(key2, value2){  html\_option +='<option value="'+ value2.value +'">' + value1.service + '-' + value1.description + '- Rp.' +value2.value+ '</option>';  $('select[name="shipping\_cost"]').html( html\_option);  });  loadSubOngkir();  loadtotals();  });  });  }  });  };  </script>  @endsection |

Kode Program 4. 24 *Source Code View Product Courier Service*

Kode program 4. 24 merupakan kode program untuk *view* product\_courier\_service dimana *view* ini mengatur tampilan dari halaman pengguna dapat memilih jasa kurir yang diinginkan untuk pengiriman ke alamat pengguna.

c. Controller

*Controller* yang digunakan untuk proses *add to cart* sekaligus untuk halaman utama yaitu *controller* HomeController. Berikut ini merupakan kode program untuk *controller* HomeController.

|  |
| --- |
| <?php  namespace App\Http\Controllers;  use Illuminate\Http\Request;  use Illuminate\Support\Facades\Auth;  class HomeController extends Controller  {  /\*\*  \* Create a new controller instance.  \*  \* @return void  \*/  public function \_\_construct()  {  $this->middleware(['auth','verified']);  }  /\*\*  \* Show the application dashboard.  \*  \* @return \Illuminate\Contracts\Support\Renderable  \*/  public function index()  {  return view('home');  // validasi role;  // if (Auth::user()->role = 'admin') {  // dd($user);  // }    }  } |

Kode Program 4. 25 *Source Code Home Controller*

Kode program 4. 25 merupakan kode program untuk *controller* HomeController dimana *controller* ini digunakan untuk mengatur proses transaksi pembelian langsung dari halaman detail produk serta transaksi pembelian dari *cart.*

#### 4.3.1.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi transaksi pembelian langsung maupun *add to cart* dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari *add to cart* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 27 Tampilan Halaman Detail Produk

Gambar 4. 27 merupakan halaman detail produk, dimana pengguna dapat melakukan transaksi langsung dengan cara klik pada tombol *buy now* atau memasukkan ke keranjang dengan klik tombol *add to cart*. Halaman detail produk lalu beralih ke halaman *mycart.*

|  |
| --- |
|  |

Gambar 4. 28 Tampilan Halaman *My Cart*

Gambar 4. 28 merupakan halaman *my cart* atau keranjang pengguna dimana terdapat *list* produk yang akan di-*checkout* oleh pengguna. Untuk melanjutkan proses transaksi pembelian pengguna bisa klik tombol *proceed to* *checkout.*

|  |
| --- |
|  |

Gambar 4. 29 Tampilan Halaman Detail Transaksi

Gambar 4. 29 merupakan halaman detail transaksi dimana pengguna mengisikan *billing address* berupa nama provinsi dan memilih jenis kurir untuk pengiriman produk menuju alamat tujuan.

|  |
| --- |
|  |

Gambar 4. 30 Tampilan Halaman *Courier Service*

Gambar 4. 30 merupakan halaman *courier service* dimana pengguna diminta memilih jenis layanan kurir yang diinginkan lalu memasukkan alamat pengiriman secara detail. Untuk mengakhiri proses transaksi pengguna dapat klik tombol *buy now.*

### 4.3.2 Membatalkan transaksi

Proses transaksi kedua yang akan diterapkan yaitu proses pembatalan transaksi pembelian dimana *user* dapat membatalkan transaksi pembelian produk setelah di-*checkout* sebelum waktu kadaluarsanya selesai. Berikut ini merupakan *source code* dan tampilan dari transaksi pembatalan transaksi.

#### 4.3.2.1 Source Code

*Source code* dari transaksi pembatalan transaksi pembelian produk dimana pengguna dapat membatalkan transaksi yang masih berjalan (belum dibayarkan) dengan menggunakan metode MVC sebagai berikut.

a. Model

Model untuk untuk proses membatalkan transaksi yaitu model Transaction\_Detail. Berikut merupakan kode program untuk model Transaction\_Detail untuk proses membatalkan transaksi pada sistem *e- commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class TransactionDetails extends Model  {  use HasFactory;  protected $table = 'transaction\_details';  protected $guarded = [];  public function Transaction(){  return $this->belongsTo(Transactions::class);  }  public function Product(){  return $this->belongsTo(Products::class);  }  } |

Kode Program 4. 26 *Source Code Transaction Detail*

Kode program 4. 26 merupakan kode program untuk model Transaction\_Detail dimana mendefinisikan proses membatalkan transaksi yang dilakukan oleh pengguna selama waktu *expired* masih berjalan.

b. View

*View* yang digunakan dalam proses pembatalan transaksi yaitu *view* payment\_confirmation dimana dalam *view* tersebut dapat melihat *list* produk yang telah di*checkout* dalam suatu transaksi, bisa mengunggah bukti pembayaran, dan juga membatalkan transaksi.

|  |
| --- |
| <main class="site-main main-container no-sidebar">  <div class="container">  <div class="bayar">  <h1 class="text-center text-primary font-weight-bolder">  {{"Rp " . number\_format($transaksi->total,2,',','.')}}  <br>  <small>Transfer To : 0988465775</small>  <hr>  </h1>  </div>  <div class="time text-center">  @if ($transaksi->status == 'canceled')  <h2>Transaksi Dibatalkan</h2>  @elseif ($transaksi->status == 'verified')  <h2>Transaksi Terverifikasi, Barang Akan Segera Dikirim</h2>  @elseif ($transaksi->proof\_of\_payment && $transaksi->status == 'delivered')  <h2> Barang Sudah Dikirim</h2>  <form action="{{Route('verif-barang-diterima',['transactions' => $transaksi->id])}}" method="POST"  enctype="multipart/form-data">  @method('put')  @csrf |

Kode Program 4. 27 *Source Code View Payment Confirmation*

Kode program 4. 27 merupakan kode program untuk *view* Payment\_Confirmation dimana *view* ini mengatur tampilan dari halaman konfirmasi pembayaran. *View* ini juga dapat digunakan oleh pengguna untuk membatalkan pesanan yang waktu *expired*nya masih berjalan.

c. Controller

*Controller* yang digunakan untuk proses membatalkan transaksi yang sedang berjalan (waktu *expired*nya masih berlangsung) yaitu *controller* HomeController. Berikut ini akan ditampilkan bagian dari *controller* tersebut yang mengatur pembatalan transaksi.

|  |
| --- |
| <?php  namespace App\Http\Controllers;  use App\Models\TransactionDetails;  use App\Models\Transactions;  use App\Models\Carts;  use Illuminate\Http\Request;  use Illuminate\Support\Facades\DB;  use Illuminate\Support\Facades\Auth;  class TransactionDetailsController extends Controller  {  /\*\*  \* Display a listing of the resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function index($transactions, Request $request)  {  $id = decrypt($transactions);  $transaksi = Transactions::with('TransactionDetails')->where('id', '=', $id)->get()->first();  $transaksidetails = TransactionDetails::with('Transaction')  ->join('transactions', 'transaction\_details.transaction\_id', '=', 'transactions.id')  ->where('transactions.id', '=', $id)  ->select(  DB::raw('SUM(transaction\_details.selling\_price \* transaction\_details.qty) as price'),  'transaction\_details.product\_id',  'transaction\_details.qty as quantity',  )  ->groupBy('transaction\_details.product\_id', 'transaction\_details.qty')  ->get();  $price = 0;  $total = 0;  $carts = Carts::where('user\_id', "=", Auth::user()->id)->where('status', '=', 'notyet')->get();  foreach ($carts as $cart) {  foreach ($cart->products->discounts as $diskon) {  if (date('Y-m-d') >= $diskon->start && date('Y-m-d') < $diskon->end) {  $price = $cart->products->price - ($diskon->percentage / 100 \* $cart->products->price);  }  }  if ($price == 0) {  $total = $total + ($cart->products->price \* $cart->qty);  } else {  $total = $total + ($price \* $cart->qty);  }  // dd($cart->qty);  }  $notificationId = request('p'); //id notif dari form  $userUnreadNotification = auth()->user()  ->unreadNotifications  ->where('id', $notificationId)  ->first();  // dd($notificationId);  if ($userUnreadNotification) {  $userUnreadNotification->update(['read\_at' => now()]);  }  return view('detail-transaksi', compact('transaksi', 'transaksidetails', 'carts', 'total'));  }  /\*\*  \* Show the form for creating a new resource.  \*  \* @return \Illuminate\Http\Response  \*/  public function create()  {  //  }  /\*\*  \* Store a newly created resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @return \Illuminate\Http\Response  \*/  public function store(Request $request)  {  //  }  /\*\*  \* Display the specified resource.  \*  \* @param \App\Models\TransactionDetails $transactionDetails  \* @return \Illuminate\Http\Response  \*/  public function show(TransactionDetails $transactionDetails)  {  //  }  /\*\*  \* Show the form for editing the specified resource.  \*  \* @param \App\Models\TransactionDetails $transactionDetails  \* @return \Illuminate\Http\Response  \*/  public function edit(TransactionDetails $transactionDetails)  {  //  }  /\*\*  \* Update the specified resource in storage.  \*  \* @param \Illuminate\Http\Request $request  \* @param \App\Models\TransactionDetails $transactionDetails  \* @return \Illuminate\Http\Response  \*/  public function update(Request $request, TransactionDetails $transactionDetails)  {  //  }  /\*\*  \* Remove the specified resource from storage.  \*  \* @param \App\Models\TransactionDetails $transactionDetails  \* @return \Illuminate\Http\Response  \*/  public function destroy(TransactionDetails $transactionDetails)  {  //  }  } |

Kode Program 4. 28 *Source Code* Pembatalan Transaksi

Kode program 4. 28 merupakan kode program bagian pembatalan transaksi pada *controller* HomeController yang telah dijabarkan pada kode program 4. 27 yaitu pada fungsi cancelOrder. Dimana status *order* akan otomatis diganti menjadi *canceled.*

#### 4.3.2.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi proses pembatalan transaksi yang masih berjalan dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari proses pembatalan transaksi yang berjalan pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 31 Tampilan Pembatalan Transaksi

Gambar 4. 31 merupakan bagian pembatalan transaksi dalam halaman detail transaksi dimana pengguna dapat melihat waktu *expired* sebuah transaksi yang belum dibayar, mengunggah bukti pembayaran dan juga membatalkan transaksi. Pembatalan transaksi dapat dilakukan oleh pengguna dengan klik tombol merah bertuliskan “*cancel order*”.

### 4.3.4 Upload Bukti Pembayaran

Proses transaksi keempat yang akan diterapkan yaitu proses *upload* bukti pembayaran dimana *user* dapat mengunggah bukti pembayaran atas transaksi pembelian yang telah dilakukan. Selanjutnya pengguna tinggal menunggu bukti pembayaran tersebut di verifikasi oleh *admin*. Berikut ini merupakan *source code* dan tampilan proses *upload* bukti pembayaran.

#### 4.3.4.1 Source Code

*Source code upload* bukti pembayaran yang dapat dilihat pada halaman detail transaksi diterapkan dengan menggunakan metode MVC. Berikut merupakan *source code upload* bukti pembayaran dari transaksi pembelian produk pada suatu transaksi pembelian produk.

a. Model

Model untuk untuk proses *upload* bukti pembayaran yaitu model Transaction\_Detail. Berikut merupakan kode program untuk model Transaction\_Detail untuk mengunggah bukti pembayaran dari suatu transaksi pada sistem *e-commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class TransactionDetails extends Model  {  use HasFactory;  protected $table = 'transaction\_details';  protected $guarded = [];  public function Transaction(){  return $this->belongsTo(Transactions::class);  }  public function Product(){  return $this->belongsTo(Products::class);  }  } |

Kode Program 4. 29 *Source Code Transaction Detail*

Kode program 4. 29 merupakan kode program untuk model Transaction\_Detail dimana mendefinisikan proses *upload* bukti pembayaran dari suatu transaksi di sistem *e-commerce.*

b. View

*View* yang digunakan dalam proses *upload* bukti pembayaran yaitu *view* payment\_confirmation dimana dalam *view* tersebut dapat melihat *list* produk yang telah di*checkout* dalam suatu transaksi, melihat sisa waktu pembayaran, mengunggah bukti pembayaran, dan membatalkan transaksi. Berikut ini adalah potongan kode program *view* payment\_confirmation untuk mengunggah bukti pembayaran.

|  |
| --- |
| <div id="modalBukti" class="modal fade">  <div class="modal-dialog" role="document">  <div class="modal-content">  <div class="modal-header">  <h4 class="modal-title" id="title-name">Bukti Pembayaran</h4>  </div>  <div class="modal-body">  <img class="w-100" src="{{$transaction->image}}" alt="">  </div>  </div><!-- /.modal-content -->  </div>  </div><!-- /.modal-dialog --> |

Kode Program 4. 30 *Source Code Upload* Bukti Pembayaran

Kode program 4. 30 merupakan bagian *upload* bukti pembayaran pada kode program *view* Payment\_Confirmation, dimana *view* ini mengatur tampilan dari halaman konfirmasi pembayaran. *View* ini juga dapat digunakan oleh pengguna untuk mengunggah bukti pembayaran dari suatu transaksi pembelian produk.

c. Controller

*Controller* yang digunakan untuk proses *upload* bukti pembayaran dari suatu transaksi pembelian produk yaitu *controller* HomeController. Kode program untuk *controller* HomeController. Berikut ini akan ditampilkan bagian dari *controller* tersebut yang mengatur proses *upload* bukti pembayaran.

|  |
| --- |
| if ($request->submit == 'Verif') {  if(!$transaction->proof\_of\_payment){  return redirect()->back()->with('warning', 'Bukti Pembayaran Belum Diupload');  }else{  $transaction->status = 'verified'; |

Kode Program 4. 31 *Source Code Upload* Bukti Pembayaran

Kode program 4. 31 merupakan kode program bagian *upload* bukti pembayaran sebuah transaksi pada *controller* HomeController, yaitu pada fungsi ProofOfPayment.

#### 4.3.4.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi *upload* bukti pembayaran sebuah transaksi yang belum dibayarkan dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari halaman *upload* bukti pembayaran yang berjalan pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 32 Tampilan *Upload* Bukti Pembayaran

Gambar 4. 32 merupakan bagian *upload* bukti pembayaran dalam halaman detail transaksi dimana pengguna dapat mengunggah sebuah gambar bukti pembayaran dengan cara *klik* tombol *choose file*. Selain itu pengguna dapat melihat waktu sebelum transaksi yang belum terbayar ini *expire* dan dapat membatalkan pesanan.

|  |
| --- |
|  |

Gambar 4. 33 Tampilan *Choose File*

Gambar 4.33 menunjukkan sebuah *windows* yang muncul ketika pengguna telah menekan tombol *choose file*. Pengguna lalu memilih *file* bukti pembayaran yang tersimpan di komputernya lalu klik *open.*

|  |
| --- |
|  |

Gambar 4. 34 Tampilan Bukti Pembayaran yang Telah di *Upload*

Gambar 4. 34 menunjukkan halaman baru yang menunjukkan gambar bukti pembayaran yang telah diunggah pengguna. Berikutnya pengguna tinggal menunggu bukti tersebut diverifikasi oleh *admin*.

### 4.3.5 Status Transaksi

Proses transaksi kelima yang akan diterapkan yaitu melihat status transaksi pembelian yang telah di *checkout*. Pengguna dapat melihat status transaksinya apakah belum diverifikasi, telah diverifikasi, telah terkirim, atau dibatalkan. Berikut ini merupakan *source code* dan tampilan status transaksi. Pada sisi admin, admin dapat melihat *list* transaksi yang dibuat di sistem *e-commerce* lalu dapat melakukan perubahan status transaksi.

#### 4.3.5.1 Source Code

*Source code* menampilkan status transaksi yang dapat dilihat pada halaman *my order* serta perubahan status transaksi yang hanya bisa dilakuakn oleh *admin* diterapkan dengan menggunakan konsep MVC. Berikut merupakan *source code* menampilkan status transaksi yang pernah dibuat oleh seorang pengguna dan mengubah status transaksi oleh *admin*.

a. Model

Model untuk untuk menampilkan status transaksi yaitu model Transaction. Berikut merupakan kode program untuk model Transaction untuk mendefinisikan *list* transaksi yang pernah dibuat oleh seorang pengguna beserta statusnya pada sistem *e-commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Transactions extends Model  {  use HasFactory;  protected $guarded = ['harga\_diskon'];  public function TransactionDetails(){  return $this->hasMany(TransactionDetails::class,'transaction\_id');  }  public function product(){  return $this->belongsToMany(Products::class, 'transaction\_details', 'transaction\_id', 'product\_id')->withPivot('id');  }  public function courier(){  return $this->belongsTo(Couriers::class, 'courier\_id', 'id');  }  public function user(){  return $this->belongsTo(User::class);  }  public function getImageAttribute()  {  return $this->proof\_of\_payment ? asset('storage/' . $this->proof\_of\_payment) : asset('assets/img/default-thumbnail.jpg');  }  } |

Kode Program 4. 32 Tampilan Model *Transaction*

Kode program 4. 32 merupakan kode program untuk model Transaction dimana mendefinisikan *list* transaksi yang pernah dibuat oleh seorang pengguna beserta statusnya pada sistem *e-commerce*. Model yang digunakan oleh *admin* untuk mengubah status transaksi adalah model Transaction yang telah dijabarkan pada kode program 4. 32 di atas.

b. View

*View* yang digunakan untuk menampilkan status transaksi yaitu *view* myorder\_list dimana dalam *view* tersebut dapat melihat *list* transaksi yang pernah dilakukan seorang pengguna beserta statusnya. Sementara *view* yang digunakan admin untuk melihat *list* transaksi yang masuk yaitu transaction.home sementara *view* yang digunakan untuk mengganti status transaksi yaitu *view* transaction.show. Berikut ini adalah kode program dari *view* myorder\_list untuk melihat status transaksi, *view* transaction.home dan transaction.show untuk melihat *list* transaksi dan mengubah status transaksi oleh *admin.*

|  |
| --- |
| </div>  <div class="time text-center">  @if ($transaksi->status == 'canceled')  <h2>Transaksi Dibatalkan</h2>  @elseif ($transaksi->status == 'verified')  <h2>Transaksi Terverifikasi, Barang Akan Segera Dikirim</h2>  @elseif ($transaksi->proof\_of\_payment && $transaksi->status == 'delivered')  <h2> Barang Sudah Dikirim</h2>  <form action="{{Route('verif-barang-diterima',['transactions' => $transaksi->id])}}" method="POST"  enctype="multipart/form-data">  @method('put')  @csrf  <button type="submit">Konfirmasi Barang Diterima</button>  </form> |

Kode Program 4. 33 Tampilan *My Order List*

Kode program 4. 33 merupakan kode program *view* myorder\_list. *View* ini menampilkan seluruh transaksi yang pernah dibuat oleh seorang pengguna beserta statusnya.

|  |
| --- |
| @extends('/admin/layouts/master',['title'=>'Transaction Detail'])  @section('content')  <section role="main" class="content-body">  <header class="page-header">  <h2>Admin</h2>  <div class="right-wrapper text-right">  <ol class="breadcrumbs pr-3">  <li>  <a href="/admin">  <i class="fas fa-home"></i>  </a>  </li>  <li>  <a href="{{Route('transaksi-admin')}}">  <span>Transaction</span>  </a>  </li>  <li><span>Transaction Detail</span></li>  </ol>    </div>  </header>  <!-- start: page -->  <div class="">  <div class="col">  <section class="card">  @include('admin/layouts/notif') |

Kode Program 4. 34 Tampilan *Transaction Home*

Kode program 4. 34 merupakan kode program *view* transaction.home. *View* ini menampilkan seluruh transaksi yang pernah dibuat oleh pengguna sistem *e-commerce* pada halaman *admin*, dimana *admin* dapat melihat status dari transaksi dan melakukan perubahan status pada halaman *view* transaction.show berikut.

|  |
| --- |
| </div>  <div class="modal-body">  <h4>Apakah Anda yakin ingin Mengupdate status transaksi ini?</h4>  <form action="{{Route('update-transaksi',['transaction'=>$transaction->id])}}" method="POST"  class="form-group" enctype="multipart/form-data">  @csrf  @method('put')  <input id="submit" type="submit" name="submit" value="" class="float-right btn btn-dark mt-3">  </div>  </form>  </div> |

Kode Program 4. 35 Tampilan *Transaction Show*

Kode program 4. 35 merupakan kode program *view* transaction.show. *View* ini menampilkan produk yang dibeli dalam suatu transaksi dan *proof of payment* (jika sudah dibayarkan). *Admin* dapat melakukan perubahan status transaksi melalui *view* ini.

c. Controller

*Controller* yang digunakan untuk melihat status transaksi pembelian produk yaitu *controller* HomeController. Sementara *controller* yang mengatur tampilan *list* transaksi di halaman *admin* dan proses mengubah status transaksi oleh *admin* yaitu TransactionController. Berikut ini akan ditampilkan bagian dari *controller* tersebut yang mengatur *my order* yang menampilkan seluruh transaksi dan statusnya serta TransactionController yang mengatur tampilan transaksi di halaman *admin* dan mengubah status transaksi oleh *admin.*

|  |
| --- |
| public function cancel(Transactions $transactions)  {  $transactions->status = 'canceled';  $transactions->update();  return redirect()->back();  }  public function verifbarang(Transactions $transactions)  {  $detailTrans = TransactionDetails::where('transaction\_id', $transactions->id)->get();  // dd($detailTrans);  foreach ($detailTrans as $detail) {  $products = Products::where('id', $detail->product\_id)->get();  foreach ($products as $key => $value) {  Products::where('id', "=", $detail->product\_id)->update([  'stock' => $value->stock - $detail->qty,  ]);  } |

Kode Program 4. 36 Tampilan *List* Transaksi

Kode program 4. 36 merupakan kode program bagian *list* transaksi yang pernah dilakukan seorang pengguna beserta statusnya pada *controller* HomeController, yaitu pada fungsi myOrder.

|  |
| --- |
| }  // dd($request->submit);  $transaction->update();  $user = User::find($transaction->user\_id);  // dd($user);  $user->notify(new UserNotification("<a class='submit-form' data-submits='' href ='/detail-transaksi/" . encrypt($transaction->id) . "'>Status Transaksi Telah Dirubah Ke " . $request->submit . " oleh admin</a>"));  return redirect(Route('transaksi-admin'))->with('success', 'Transaksi Berhasil Diupdate');  } |

Kode Program 4. 37 Tampilan *Transaction Controller*

Kode program 4. 37 merupakan kode program *controller* TransactionController yang mengatur tampilan *list* transaksi pada halaman admin dan mengatur proses perubahan status transaksi oleh *admin.*

#### 4.3.5.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi melihat status transaksi-transaksi yang sudah dibuat atau di*checkout* oleh seorang pengguna dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari halaman *my order* yang berjalan serta tampilan *list* transaksi di halaman *admin* dan tampilan proses perubahan status transaksi oleh *admin* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 35 Tampilan Halaman *My Order*

Gambar 4. 35 merupakan tampilan dari halaman *my order* yang berisi seluruh transaksi yang telah dilakukan seorang pengguna lengkap dengan statusnya (*verified, unverified, delivered,* atau *canceled*).

|  |
| --- |
|  |

Gambar 4. 36 Tampilan Halaman *Transaction*

Gambar 4. 36 merupakan tampilan dari halaman *transaction* pada menu *Master Data Admin*, dimana ditampilkan *list* transaksi di sebuah sistem *e-commerce* lengkap dengan statusnya. *Admin* dapat klik pada tombol *details* untuk melihat detail transaksi dan mengubah status transaksinya.

|  |
| --- |
|  |

Gambar 4. 37 Tampilan Halaman Detail *Transaction*

Gambar 4. 37 merupakan tampilan dari halaman detail *transaction* pada menu *Master Data Admin*, dimana ditampilkan detail dari sebuah transaksi. *Admin* dapat klik pada tombol *change status* untuk melakukan perubahan status transaksi.

|  |
| --- |
|  |

Gambar 4. 38 Tampilan *Change Status*

Gambar 4. 38 merupakan tampilan dari sebuah jendela yang muncul setelah *admin* klik pada tombol *change status*. Pada jendela tersebut *admin* dapat mengganti status transaksi misalnya yang awalnya *verified* menjadi *delivered*, untuk menyimpan perubahan *admin* klik pada tombol *submit*.

|  |
| --- |
|  |

Gambar 4. 39 Tampilan Halaman Transaksi

Gambar 4. 39 merupakan tampilan dari halaman transaksi. Gambar ini menunjukkan perubahan status yang terjadi pada transaksi keempat dalam *list* transaksi tersebut telah berhasil yaitu dari status awal *verified* menjadi *delivered.*

### 4.3.6 Review dan Response

Proses transaksi keenam yang akan diterapkan yaitu menambahkan *review* (untuk *user*) pada halaman produk dan memberikan respon atas *review* pengguna oleh *admin*. Berikut ini merupakan *source code* dan tampilan *review* dan *response.*

#### 4.3.6.1 Source Code

*Source code* menambahkan *review* (untuk *user*) pada halaman produk dan memberikan respon atas *review* pengguna oleh *admin* diterapkan dengan menggunakan metode MVC. Berikut merupakan *source code* menambahkan *review* (untuk *user*) pada halaman produk dan memberikan respon atas *review* pengguna oleh *admin*.

a. Model

Model untuk untuk menambahkan *review* dari *user* untuk sebuah produk yaitu model Product\_review. Sementara model untuk *admin* meningalkan respon dari *review user* yaitu model Response.Berikut merupakan kode program untuk kedua model tersebut.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class ReviewProducts extends Model  {  use HasFactory;  protected $table = 'product\_reviews';  protected $guarded = [];  public function Product(){  return $this->belongsTo(Products::class);  }  public function User(){  return $this->belongsTo(User::class);  }  public function response(){  return $this->hasMany(Response::class, 'review\_id', 'id');  }  } |

Kode Program 4. 38 *Source Code Product Review*

Kode program 4. 38 merupakan kode program untuk model Product\_review dimana pengguna dapat menambahkan *review* atas sebuah produk dalam sistem *e-commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Response extends Model  {  use HasFactory;  protected $table = 'response';  protected $guarded = [];  public function admin(){  return $this->belongsTo(Admin::class, 'admin\_id', 'id');  }  } |

Kode Program 4. 39 *Source Code Response*

Kode program 4. 39 merupakan kode program untuk model Response dimana *admin* dapat menambahkan respon atas *review user* yang telah masuk pada sistem *e-commerce.*

b. View

*View* yang digunakan untuk *user* menambahkan *review* pada sebuah produk yaitu *view* detail\_product, sementara *view* yang digunakan untuk menampilkan halaman untuk *admin* mengirimkan *list review* pengguna yaitu *view* response.home serta *view* response.show untuk menambahkan respon atas *review* pengguna. Berikut ini adalah kode program dari *view* myorder\_list untuk melihat status transaksi.

|  |
| --- |
| <li class="reviews\_tab" id="tab-title-reviews" role="tab" aria-controls="tab-reviews">  <a href="#tab-reviews">Reviews ({{$products->reviews->count()}})</a>  </li>  </ul>  <div class="BLitzz-Tabs-panel BLitzz-Tabs-panel--description panel entry-content BLitzz-tab"  id="tab-description" role="tabpanel" aria-labelledby="tab-title-description">  <h2>Description</h2>  {!!$products->description!!}  </div>  <div class="BLitzz-Tabs-panel BLitzz-Tabs-panel--reviews panel entry-content BLitzz-tab"  id="tab-reviews" role="tabpanel" aria-labelledby="tab-title-reviews">  <div id="reviews" class="BLitzz-Reviews">  <div class="comments-list text-center text-md-left">  @if (!$products->reviews->count())  <div class="d-flex justify-content-center">  <div class="row mb-5">  <p><strong>Belum ada review produk.</strong></p>  </div>  </div>  @else  @foreach ($products->reviews as $item)  <!-- First row --> |

Kode Program 4. 40 *Source Code* Detail *Product*

Kode program 4. 40 merupakan kode program *view* detail\_product. *View* ini menampilkan semua *review* pada sebuah produk dan meninggalkan sebuah *review* pada halaman produk di sistem *e-commerce.*

|  |
| --- |
| <h5 class="font-weight-bold text-dark">product review</h5>  </header>  <div class="card-body">  <table class="table table-striped">  <thead>  <tr>  <th>User</th>  <th>Rate</th>  <th>Review</th>  <th>Response</th>  <th class="text-center">Action</th>  </tr>  </thead>  <tbody>  @foreach ($product->reviews as $review)  <tr>  <td>{{$review->user->name}}</td>  <td>  @for ($i = 0; $i < $review->rate; $i++)  <i class="fas fa-star text-warning"></i>  @endfor  </td>  <td>{{$review->content}}</td>  <td>  @foreach($responses as $response)  @if($review->id == $response->review\_id)  {{ $response->content }},  @endif  @endforeach  </td> |

Kode Program 4. 41 *Source Code Response Home*

Kode program 4. 41 merupakan kode program *view* response.home. *View* ini menampilkan seluruh *review* yang dikirimkan oleh *user* dalam suatu sistem *e- commerce.*

|  |
| --- |
| <td>{{$review->content}}</td>  <td>  @foreach($responses as $response)  @if($review->id == $response->review\_id)  {{ $response->content }},  @endif  @endforeach  </td>  <td class="text-center">  <button data-id="{{$review->id}}" data-nama="{{$review->user->name}}"  class="respond btn btn-sm btn-success" data-toggle="modal"  data-target="#respondform">Balas</button> |

Kode Program 4. 42 *Source Code Response Show*

Kode program 4. 42 merupakan kode program *view* response.show. *View* ini menampilkan detail *review* dan mengirimkan respon atas *review* barang oleh pengguna tersebut.

c. Controller

*Controller* yang digunakan untuk menambahkan *review* suatu produk oleh seorang *user* yaitu *controller* HomeController. Sementara *controller* untuk menambahkan respon dari *review user* oleh *admin* yaitu *controller* ResponseController. Berikut ini akan ditampilkan bagian dari *controller* tersebut yang mengatur proses menambahkan *review* produk oleh pengguna dan *controller* yang mengatur respon *admin.*

|  |
| --- |
| $review = New ReviewProducts;    $review->product\_id = $request->id;  $review->rate = $request->rate;  $review->content = $request->content;  $review->user\_id = Auth::user()->id;  $review->save();  // $admin = Admin::where('role', 'super admin')->get();  $admin = Admin::get();  // dd($admin);  $product = Products::find($review->product\_id); |

Kode Program 4. 43 *Source Code* Tambah *Response*

Kode program 4. 43 merupakan kode program menambahkan sebuah *review* atas sebuah produk oleh pembeli atau pengguna pada *controller* HomeController yaitu melalui fungsi insertReview.

|  |
| --- |
| public function show(ReviewProducts $reviewProducts)  {  //  } |

Kode Program 4. 44 *Source Code List Review*

Kode program 4. 44 merupakan kode program yang mengatur *list review* yang dapat direspon oleh *admin* pada *dashboard* admin yaitu *controller* ResponseController. Fungsi yang digunakan untuk menambahkan respon yaitu fungsi giveResponse.

#### 4.3.6.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi menambahkan *review* produk oleh pengguna dan memberikan respond dari *review* pengguna oleh *admin* dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari halaman menambahkan *review* dan memberikan respon pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 40 Tampilan Halaman *Review*

Gambar 4. 40 merupakan tampilan dari *window review* dimana pengguna dapat menambahkan *review* baru untuk produk dengan memasukkan *rating* serta komentar lalu klik *submit* untuk menambahkan *review.*

|  |
| --- |
|  |

Gambar 4. 41 Tampilan Halaman *Review* Disimpan

Gambar 4. 41 merupakan tampilan ketika *review* yang baru saja ditambahkan berhasil disimpan. *Review* terbaru ditambahkan pada bagian terbawah dari *list review* produk.

|  |
| --- |
|  |

Gambar 4. 42 Tampilan Halaman *Review* pada *Dashboard Admin*

Gambar 4. 42 merupakan tampilan halaman ulasan pembeli pada *dashboard* admin dimana admin dapat melihat *list review* pembeli yang masuk. Admin dapat melihat detail ulasan dengan klik tombol *detail.*

|  |
| --- |
|  |

Gambar 4. 43 Tampilan Halaman Detail Ulasan

Gambar 4. 43 merupakan tampilan halaman detail ulasan pembeli pada *dashboard admin*. *Admin* dapat menambahkan respon dari *review* tersebut dengan klik *leave response.*

|  |
| --- |
|  |

Gambar 4. 44 Tampilan *Window Leave Response*

Gambar 4. 44 merupakan tampilan *window* yang muncul setelah admin klik tombol *leave response. Admin* dapat memberikan tanggapan atas *review* pengguna lalu klik tombol *submit* untuk mengirimkan respon tersebut.

## 4.4 Report

Modul keempat dari praktikum pemrograman internet yaitu pembuatan sistem notifikasi dan *report* pada rancangan sistem *e-commerce* dimana sistem dapat memberi notifikasi untuk pembeli, *admin*, dan memberi laporan transaksi yang ditampilkan pada *dashboard admin* meliputi jumlah transaksi per bulan dan per tahun serta suatu grafik yang menunjukkan naik atau turunnya penjualan per bulan.

### 4.4.1 Notifikasi Pembeli

*Report* atau pelaporan pertama yang akan diimplementasikan yaitu notifikasi kepada pembeli atau *user*. Notifikasi yang diterima oleh pengguna yaitu terjadi perubahan status transaksi misalnya dari *unverified* menjadi *verified* ketika bukti pembayaran transaksi sudah diverifikasi *admin* atau dari *verified* menjadi *delivered* ketika produk telah dikirimkan ke alamat pembeli. Notifikasi lainnya yang didapatkan oleh pembeli adalah ketika *admin* merespon atas komentar pembeli. Berikut ini merupakan *source code* dan tampilan dari notifikasi pembeli pada sistem *e-commerce.*

#### 4.4.1.1 Source Code

*Source code* untuk notifikasi pembeli diterapkan dengan menggunakan metode MVC. Berikut merupakan *source code* menampilkan notifikasi pembeli dalam sebuah sistem *e-commerece*.

a. Model

Model untuk untuk menampilkan notifikasi pembeli yaitu model User\_notification. Berikut merupakan kode program untuk model User\_notification untuk mendefinisikan notifikasi pembeli atau seorang pengguna beserta pada sistem *e-commerce*.

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  class UserNotification extends Model  {  protected $table = 'user\_notifications';  protected $guarded = [];  } |

Kode Program 4. 45 *Source Code User Notification*

Kode program 4. 45 merupakan kode program untuk model User\_notification dimana mendefinisikan notifikasi pembeli atau seorang *user* pada sistem *e-commerce.*

b. View

*View* yang digunakan untuk menampilkan notifikasi pembeli atau *user* yaitu *view* user\_page.notification dimana dalam *view* tersebut dapat melihat *list* notifikasi pembeli atau *user* di sebuah sistem *e-commerce*. Berikut ini adalah kode program dari *view* user\_page.notification untuk melihat notifikasi pembeli.

|  |
| --- |
| iv class="time text-center">  @if ($transaksi->status == 'canceled')  <h2>Transaksi Dibatalkan</h2>  @elseif ($transaksi->status == 'verified')  <h2>Transaksi Terverifikasi, Barang Akan Segera Dikirim</h2>  @elseif ($transaksi->proof\_of\_payment && $transaksi->status == 'delivered')  <h2> Barang Sudah Dikirim</h2>  <form action="{{Route('verif-barang-diterima',['transactions' => $transaksi->id])}}" method="POST"  enctype="multipart/form-data">  @method('put')  @csrf  <button type="submit">Konfirmasi Barang Diterima</button>  </form>  @elseif ($transaksi->status == 'success')  <h2> Barang Sudah Diterima, Transaksi Success</h2>  @elseif ($transaksi->status == 'expired')  <h2> TRANSAKSI EXPIRED</h2>  @else  @if (!$transaksi->proof\_of\_payment && $transaksi->status == 'unverified'|| $transaksi->status == 'failed')  <div id="countdown"> </div>  <form action="{{Route('cancel',['transactions' => $transaksi->id])}}" method="POST"  enctype="multipart/form-data">  @method('put')  @csrf  <button type="submit" onclick="return confirm('Yakin ingin Membatalkan Pesanan?')">Cancel Transaction</button>  </form>  @else  <h2>Bukti Pembayaran Diterima, silahkan tunggu admin memverifikasi bukti pembayaran</h2>  @endif  @endif  </div>  <div class="row pt-5">  @if ($transaksi->status == 'unverified' && $transaksi->proof\_of\_payment == null || $transaksi->status ==  'failed' ) |

Kode Program 4. 46 *Source Code View User Page Notification*

Kode program 4. 46 merupakan kode program *view* user\_page.notification. *View* ini menampilkan notifikasi yang diterima oleh seorang pembeli atau *user* dalam sistem *e-commerce*, notifikasi itu berupa perubahan status transaksi yang pernah dilakukan. Notifikasi lainnya yang didapatkan oleh pembeli adalah ketika admin merespon atas komentar pembeli.

c. Controller

*Controller* yang digunakan untuk melihat notifikasi pembeli yaitu *controller* NotifyUser. *Controller* lainnya yang digunakan dalam penerapan notifikasi pembeli yaitu *controller* TransactionController untuk notifikasi perubahan status transaksi, serta *controller* ResponseController untuk notifikasi respon dari *admin*. Berikut ini akan ditampilkan kode program dari *controller* NotifyUser, TransactionController, dan ResponseController yang mengatur seluruh notifikasi untuk pembeli atau *user* di suatu sistem *e- commerce.*

|  |
| --- |
| public function \_\_construct()  {  $this->middleware('auth');  $this->middleware('signed')->only('verify');  $this->middleware('throttle:6,1')->only('verify', 'resend');  }  } |

Kode Program 4. 47 *Source Code Notify User*

Kode program 4. 47 merupakan kode program *controller* NotifyUser yang digunakan untuk mengatur seluruh notifikasi yang diterima oleh seorang pembeli atau *user* pada sistem *e-commerce.*

|  |
| --- |
| if ($request->proof\_of\_payment) {  $gambar = $request->file('proof\_of\_payment');  $urlgambar = $gambar->storeAs("img/bukti", md5('Bukti' . $transactions->id . microtime()) . '.' . $gambar->extension());  $transactions->proof\_of\_payment = $urlgambar;  $transactions->status = 'unverified';  $transactions->update(); |

Kode Program 4. 48 *Source Code* Notifikasi Perubahan Status Pesanan

Kode program 4. 48 merupakan kode program bagian pemberian notifikasi perubahan status pesanan pada *controller* TransactionController yang sebelumnya telah dijabarkan.

#### 4.4.1.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi melihat seluruh notifikasi yang diterima oleh seorang pengguna dalam rancangan sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari halaman notifikasi pembeli pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
| $user->name = $request->name;  if ($user->email != $request->email) {  $user->email = $request->email;  $user->email\_verified\_at = null;  $user->sendEmailVerificationNotification();  }  if ($request->file('profile\_image')) {  Storage::delete($user->profile\_image);  $gambar = $request->file('profile\_image');  $nama\_image = md5(now() . "\_") . $request->file('profile\_image')->getClientOriginalName();  $urlgambar = $gambar->storeAs("img/profile\_image", $nama\_image);  $user->profile\_image = $urlgambar;  }  // dd($users);  $user->update();  return redirect()->back()->with('success', 'Profile Telah Diupdate');  } |

Kode Program 4. 49 *Source Code List* Notifikasi

Gambar 4. 49 merupakan tampilan dari halaman berisi *list* notifikasi yang diterima oleh seorang *user* dalam sistem *e-commerce*. Pengguna dapat menekan tombol *mark all read* atau secara individual di kolom *action* sehingga notifikasi tersebut tidak lagi muncul pada menu *notification.*

### 4.4.2 Notifikasi Admin

*Report* atau pelaporan kedua yang akan diimplementasikan yaitu notifikasi kepada *admin*. Notifikasi yang diterima oleh *admin* yaitu ketika ada transaksi baru yang masuk atau ketika ada bukti pembayaran atas sebuah transaksi yang diunggah oleh pengguna. Notifikasi lainnya yaitu ketika seorang pengguna mengirimkan *review* atau komen pada sebuah produk. Berikut ini merupakan *source code* dan tampilan dari notifikasi *admin* pada sistem *e-commerce.*

#### 4.4.2.1 Source Code

*Source code* untuk notifikasi *admin* diterapkan dengan menggunakan metode MVC. Berikut merupakan *source code* menampilkan notifikasi *admin* dalam sebuah sistem *e-commerece.*

a. Model

Model untuk untuk menampilkan notifikasi *admin* yaitu model admin\_notification. Berikut merupakan kode program untuk model admin\_notification untuk mendefinisikan notifikasi *admin* pada sistem *e- commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class AdminNotification extends Model  {  use HasFactory;  protected $table = 'admin\_notifications';  protected $guarded = [];  } |

Kode Program 4. 50 *Source Code Admin Notification*

Kode program 4. 50 merupakan kode program untuk model admin\_notification dimana mendefinisikan notifikasi *admin* pada sistem *e-commerce.*

b. View

*View* yang digunakan untuk menampilkan notifikasi *admin* yaitu *view* admin.notification dimana dalam *view* tersebut dapat melihat *list* notifikasi yang diterima *admin* di sebuah sistem *e-commerce*. Berikut ini adalah kode program dari *view* admin.notification untuk melihat notifikasi *admin.*

|  |
| --- |
| <h2 class="mb-2">Jumlah Transaksi: <span><strong  id="total-tahun">{{$transaksi\_tahun->count()}}</strong></span></h2>  <p>Unverified Transaction <span> <strong  id="unverified-tahun">{{$status\_tahun['unverified']}}</strong></span></p>  <p>Expired Transaction <span><strong id="expired-tahun">{{$status\_tahun['expired']}}</strong></span>  </p>  <p>Canceled Transaction <span><strong  id="canceled-tahun">{{$status\_tahun['canceled']}}</strong></span></p>  <p>Verified Transaction <span><strong  id="verified-tahun">{{$status\_tahun['verified']}}</strong></span></p>  <p>Delivered Transaction <span><strong  id="delivered-tahun">{{$status\_tahun['delivered']}}</strong></span></p>  <p>Success Transaction <span><strong id="success-tahun">{{$status\_tahun['success']}}</strong></span>  </p>  <h4>Total Penghasilan <span><strong id="harga-tahun">Rp  {{number\_format($status\_tahun['harga'],0,',','.')}}</strong></span></h4>  </div> |

Kode Program 4. 51 *Source Code View Admin Notification*

Kode program 4. 51 merupakan kode program *view* admin.notification. *View* ini menampilkan notifikasi yang diterima oleh seorang *admin* dalam sistem *e commerce*, notifikasi itu berupa transaksi yang masuk (yang memerlukan perubahan status dari *unverified* ke *verified*) serta ulasan pembeli yang masuk.

c. Controller

*Controller* yang digunakan untuk melihat notifikasi *admin* yaitu *controller* NotifyController. *Controller* lainnya yang mengatur notifikasi *admin* yaitu *controller* HomeController untuk menambahkan notifikasi masuknya sebuah *proof of payment* suatu transaksi dan *review* dari *user*. Berikut ini akan ditampilkan kode program dari *controller* NotifyController dan HomeController yang mengatur seluruh notifikasi untuk admin di suatu sistem *e-commerce.*

|  |
| --- |
| $transaksi = new Transactions;  $datetime = strtotime("+1 day");  $tomorrowDate = date("Y-m-d H:i:s", $datetime);  $transaksaction = $request->all();  $transaksaction['timeout'] = $tomorrowDate;  $transaksaction['total'] = $request->shipping\_cost + $total;  $transaksaction['sub\_total'] = $total;  $transaksaction['user\_id'] = Auth::user()->id;  $transaksaction['status'] = 'unverified';  $transaksaction['telp'] = $request->telp;  // dd($transaksaction);  $validator = Validator::make($transaksaction, |

Kode Program 4. 52 *Source Code Notify Controller*

Kode program 4. 52 merupakan kode program *controller* NotifyController yang digunakan untuk mengatur seluruh notifikasi yang diterima oleh seorang *admin* pada sistem *e-commerce.*

|  |
| --- |
| if ($request->submit == 'Verif') {  if(!$transaction->proof\_of\_payment){  return redirect()->back()->with('warning', 'Bukti Pembayaran Belum Diupload');  }else{  $transaction->status = 'verified'; |

Kode Program 4. 53 *Source Code* Notifikasi *Proof Of Payment*

Kode program 4. 53 merupakan potongan dari kode program *controller* HomeController. *Controller* ini digunakan untuk mengatur notifikasi ketika masuknya *proof of payment* pada sebuah transaksi dan notifikasi jika ada *review* barang yang masuk pada sistem *e- commerce.*

#### 4.4.2.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi melihat seluruh notifikasi yang diterima oleh admin sistem *e- commerce*. Berikut ini akan ditampilkan tampilan dari halaman notifikasi *admin* pada rancangan *e-commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 45 Tampilan Halaman *List Notifikasi*

Gambar 4. 45 merupakan tampilan dari halaman berisi *list* notifikasi yang diterima oleh admin dalam sistem *e-commerce*. Pengguna dapat menekan tombol *mark all read* atau secara individual di kolom *action* sehingga notifikasi tersebut tidak lagi muncul pada menu *notification.*

### 4.4.3 Laporan Transaksi

*Report* atau pelaporan ketiga yang akan diimplementasikan yaitu pelaporan transaksi pada *dashboard* admin. Laporan ini hanya bisa diakses oleh pengguna berstatus admin di *dashboard* admin. Pelaporan transaksi di antaranya pendapatan per bulan dan pertahun, jumlah transaksi per bulan dan pertahun yang digambarkan sebuah diagram serta tabel berisi rincian pendapatan dan jumlah transaksi. Berikut ini merupakan *source code* dan tampilan dari laporan transaksi pada sistem *e-commerce.*

#### 4.4.3.1 Source Code

*Source code* untuk laporan transaksi diterapkan dengan menggunakan metode MVC. Berikut merupakan *source code* menampilkan laporan transaksi dalam sebuah sistem *e-commerece.*

a. Model

Model untuk untuk menampilkan laporan transaksi yaitu model Transaction. Berikut merupakan kode program untuk model Transaction untuk mendefinisikan laporan transaksi yang hanya bisa diakses oleh *admin* pada sistem *e-commerce.*

|  |
| --- |
| <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Database\Eloquent\Model;  class Transactions extends Model  {  use HasFactory;  protected $guarded = ['harga\_diskon'];  public function TransactionDetails(){  return $this->hasMany(TransactionDetails::class,'transaction\_id');  }  public function product(){  return $this->belongsToMany(Products::class, 'transaction\_details', 'transaction\_id', 'product\_id')->withPivot('id');  }  public function courier(){  return $this->belongsTo(Couriers::class, 'courier\_id', 'id');  }  public function user(){  return $this->belongsTo(User::class);  }  public function getImageAttribute()  {  return $this->proof\_of\_payment ? asset('storage/' . $this->proof\_of\_payment) : asset('assets/img/default-thumbnail.jpg');  }  } |

Kode Program 4. 54 *Source Code Transaction*

Kode program 4. 54 merupakan kode program untuk model Transaction dimana mendefinisikan laporan transaksi bulanan dan tahunan yang hanya bisa diakses admin pada sistem *e-commerce.*

b. View

*View* yang digunakan untuk menampilkan halaman *dashboard* berisi laporan transaksi yaitu *view* admin.home dimana dalam *view* tersebut menampilkan laporan transaksi bulanan dan tahunan di sebuah sistem *e-commerce*. Berikut ini adalah kode program dari *view* admin.home untuk melihat laporan transaksi.

|  |
| --- |
| var options = {  axisX: {  interval:1,  labelMaxWidth: 180,  labelAngle: -45,  labelFontFamily:"Century Gothic"  },  title: {  text: "Grafik Jumlah Transaksi Perbulan {{date('Y')}}"  },  data: [  {  type: "column",  dataPoints: [  { label: "Januari", y: parseInt($('#bulan1').val())},  { label: "Februari", y: parseInt($('#bulan2').val())},  { label: "Maret", y: parseInt($('#bulan3').val())},  { label: "April", y: parseInt($('#bulan4').val())},  { label: "Mei", y: parseInt($('#bulan5').val())},  { label: "Juni", y: parseInt($('#bulan6').val())},  { label: "Juli", y: parseInt($('#bulan7').val())},  { label: "Agustus", y: parseInt($('#bulan8').val())},  { label: "September", y: parseInt($('#bulan9').val())},  { label: "Oktober", y: parseInt($('#bulan10').val())},  { label: "November", y: parseInt($('#bulan11').val())},  { label: "Desember", y: parseInt($('#bulan12').val())},  ]  }  ]  };    $("#chartContainer").CanvasJSChart(options);  }  </script>  <script>  function formatRupiah(angka, prefix){  var number\_string = angka.toString(),  split = number\_string.split(','),  sisa = split[0].length % 3,  rupiah = split[0].substr(0, sisa),  ribuan = split[0].substr(sisa).match(/\d{3}/gi);    // tambahkan titik jika yang di input sudah menjadi angka ribuan  if(ribuan){  separator = sisa ? '.' : '';  rupiah += separator + ribuan.join('.');  }    rupiah = split[1] != undefined ? rupiah + ',' + split[1] : rupiah;  return prefix == undefined ? rupiah : (rupiah ? 'Rp ' + rupiah : '');  }  $(document).ready(function(e){  console.log($('#bulan1').val())  $('#bulan').change(function(e){  $.ajax({  url: "{{url('/report-bulan')}}",  method: 'post',  data: {  bulan: $('#bulan').val(),  tahun: $('#tahun').val(),  },  success: function(result){  $('#total').text(result.data['total']);  $('#unverified').text(result.data['unverified']);  $('#expired').text(result.data['expired']);  $('#canceled').text(result.data['canceled']);  $('#verified').text(result.data['verified']);  $('#delivered').text(result.data['delivered']);  $('#success').text(result.data['success']);  var uang = formatRupiah(result.data['harga'],'Rp ');  $('#harga').text(uang);  }  });  });  $('#tahun').change(function(e){  $.ajax({  url: "{{url('/report-tahun')}}",  method: 'post',  data: {  \_token: $('#signup-token').val(),  bulan: $('#bulan').val(),  tahun: $('#tahun').val(),  },  success: function(result){  $('#total').text(result.data\_bulan['total']);  $('#unverified').text(result.data\_bulan['unverified']);  $('#expired').text(result.data\_bulan['expired']);  $('#canceled').text(result.data\_bulan['canceled']);  $('#verified').text(result.data\_bulan['verified']);  $('#delivered').text(result.data\_bulan['delivered']);  $('#success').text(result.data\_bulan['success']);  var uang = formatRupiah(result.data\_bulan['harga'],'Rp ');  $('#harga').text(uang);  $('#total-tahun').text(result.data['total']);  $('#unverified-tahun').text(result.data['unverified']);  $('#expired-tahun').text(result.data['expired']);  $('#canceled-tahun').text(result.data['canceled']);  $('#verified-tahun').text(result.data['verified']);  $('#delivered-tahun').text(result.data['delivered']);  $('#success-tahun').text(result.data['success']);  var uang\_tahun = formatRupiah(result.data['harga'],'Rp ');  $('#harga-tahun').text(uang\_tahun);    creteChart(result.tahun, $('#tahun').val());  }  });  });  function creteChart(tahun, ttlTahun, judul = ''){  var options = {  axisX: {  interval:1,  labelMaxWidth: 180,  labelAngle: -45,  labelFontFamily:"Century Gothic"  },  title: {  text: "Grafik Jumlah Transaksi "+judul+" Perbulan "+ttlTahun  },  data: [  {  type: "column",  dataPoints: [  { label: "Januari", y: tahun[1]},  { label: "Februari", y: tahun[2]},  { label: "Maret", y: tahun[3]},  { label: "April", y: tahun[4]},  { label: "Mei", y: tahun[5]},  { label: "Juni", y: tahun[6]},  { label: "Juli", y: tahun[7]},  { label: "Agustus", y: tahun[8]},  { label: "September", y: tahun[9]},  { label: "Oktober", y: tahun[10]},  { label: "November", y: tahun[11]},  { label: "Desember", y: tahun[12]},    ]  }  ]  };    $("#chartContainer").CanvasJSChart(options);  }  $("#status").change(function(e){  var myStatus = $('#status').val();  // console.log(myStatus);    $.ajax({  url: "{{url('/grafik')}}",  method: 'post',  data: {  status: myStatus,  tahun: $('#tahun').val(),  },  success: function(result){  creteChart(result.grafik, $('#tahun').val(), myStatus);  }  });  });  });  </script>  @endsection |

Kode Program 4. 55 *Source Code View Home Admin*

Kode program 4. 55 merupakan kode program *view* admin.home. *View* ini menampilkan halaman *dashboard admin* yang memuat laporan transaksi dalam sistem *e-commerce.*

c. Controller

*Controller* yang digunakan untuk melihat laporan transaksi pada *e- commerce* di halaman *dashboard* admin yaitu *controller* AdminController. Berikut ini akan ditampilkan kode program dari *controller* AdminController yang mengatur halaman *dashboard* admin berisi laporan transaksi di suatu sistem *e- commerce.*

|  |
| --- |
| for ($i = 1; $i <= 12; $i++) {  $tahun[$i] = Transactions::whereMonth('created\_at', '=', $i)->whereYear('created\_at', '=', $request->tahun)->count();  }  return response()->json(['success' => 'berhasil', 'data' => $status, 'data\_bulan' => $status\_bulan, 'tahun' => $tahun]);  }  public function findCountStatus($status, $bulan, $tahun, $cek)  {  if ($cek == 1) {  $count = Transactions::whereMonth('created\_at', '=', $bulan)->whereYear('created\_at', '=', $tahun)->where('status', '=', $status)->count();  } else {  $count = Transactions::whereYear('created\_at', '=', $tahun)->where('status', '=', $status)->count();  }  return $count;  }  public function grafik(Request $request)  {  // dd($request->status);  if ($request->status == 'all') {  for ($i = 1; $i <= 12; $i++) {  $grafik[$i] = Transactions::whereMonth('created\_at', '=', $i)->whereYear('created\_at', '=', $request->tahun)->count();  }  } else {  for ($i = 1; $i <= 12; $i++) {  $grafik[$i] = Transactions::whereMonth('created\_at', '=', $i)->whereYear('created\_at', '=', $request->tahun)->where('status', '=', $request->status)->count();  }  }  return response()->json(['success' => 'berhasil', 'grafik' => $grafik]);  }  public function userProfile()  {  $adminUser = Admin::where('id', Auth::guard('admin')->user()->id)->first();  return view('admin/admin\_profile/index', compact('adminUser'));  } |

Kode Program 4. 56 *Source Code Admin Controller*

Kode program 4. 56 merupakan kode program *controller* AdminController yang digunakan untuk mengatur halaman *dashboard admin* yang memuat laporan transaksi pada sistem *e-commerce.*

#### 4.4.3.2 Tampilan

Sebelumnya telah dijabarkan kode program yang digunakan dalam implementasi melihat laporan transaksi per bulan dan per tahun yang hanya dapat diakses *admin* sistem *e-commerce*. Berikut ini akan ditampilkan tampilan dari halaman *dashboard admin* yang memuat laporan transaksi pada rancangan *e- commerce* berdasarkan kode program di atas.

|  |
| --- |
|  |

Gambar 4. 46 Tampilan Halaman Laporan Transaksi

Gambar 4. 46 merupakan tampilan dari halaman *dashboard admin* yang berisi laporan transaksi bulanan dan tahunan dari sebuah sistem *e-commerce*. Laporan ini ditampilkan dalam bentuk *chart* dan rincian tabel, dan hanya dapat diakses oleh *admin*.

# BAB V PENUTUP

Bab V merupakan bagian dari laporan yang berisikan kesimpulan dan saran dari laporan praktikum Pemrograman Internet ini. Kesimpulan dan saran atas laporan praktikum ini dapat dilihat sebagai berikut.

## 5.1 Kesimpulan

Berdasarkan pembahasan yang telah dijabarkan pada pada Laporan Praktikum Pemrograman Internet serta modul-modul yang ada dapat ditarik kesimpulan, yaitu pembuatan *website e-commerce* dengan menerapkan *multi-authentication*, pemrosesan *database* CRUD pada produk, *courier*, dan *category*, implementasi proses transaksi pada rancangan sistem, dam sistem notifikasi baik untuk admin dan pengguna, serta laporan transaksi dapat diterapkan dengan menggunakan bahasa pemrograman internet seperti PHP, HTML, CSS dan lainnya dengan *framework* Laravel yang dapat diakses menggunakan *engine* XAMPP pada *localhost*.

## 5.2 Saran

Saran yang dapat diberikan setelah penyusunan Laporan Praktikum Pemrograman Internet ini yaitu *website e-commerce* ini masih memiliki sejumlah kekurangan yang perlu diadakan pengembangan lebih lanjut ke depannya untuk meningkatkan fungsionalitas dari *website*. Selain itu ke depannya *website* ini dapat di *hosting* pada layanan *hosting* supaya bisa diakses oleh siapa saja tentu saja setelah diadakan pengembangan untuk hasil *website* yang layak untuk dipergunakan massal.

# DAFTAR PUSTAKA

Aminudin. (2015). Cara Efektif Belajar Framework Laravel. Yogyakarta: Penerbit: Lokomedia

C., A. 2019. Apa Itu CSS? Pengertian dan Cara Kerjanya. Retrieved from Apa Itu CSS?: <https://www.hostinger.co.id/tutorial/apa-itu-css/>

C., A. 2019. Apa Itu GitHub? Kenali Pengertian dan Fungsinya. Retrieved from Hostinger Tutorial: <https://www.hostinger.co.id/tutorial/apa-itu-github/>

C., A. 2019. Apa Itu JavaScript? Pemahaman Dasar Mengenai JavaScript bagi Para Pemula Retrieved from Apa Itu JavaScript?: <https://www.hostinger.co.id/tutorial/apa-itu-javascript/>

C., A. 2019. Apa Itu JQuery? Retrieved from Apa Itu JQuery?: <https://www.hostinger.co.id/tutorial/apa-itu-jquery/>

Danny, U. 2016 . Tutorial CodeIgniter PDF. Retrieved from Panduan Praktis Tutorial Belajar Office dan Sistem Pakar: [http://umardanny.com/tutorial- codeigniter-pdf/](http://umardanny.com/tutorial-%20codeigniter-pdf/)

Erawan, L. 2014. Dasar-Dasar PHP. Retrieved from Sistem Informasi - Fasilkom: <https://dinus.ac.id/repository/docs/ajar/modul8teori_-_PHP.pdf>

Fajaryati, N. 2014. Pemrograman Web. Retrieved from HTML: http://staffnew.uny.ac.id/upload/198401312014042002/pendidikan/labshee t%203%20-%20HTML.pdf

K., Y. 2019. Pengertian jQuery Serta Fungsi dan Contohnya. Retrieved from JQuery adalah: <https://www.niagahoster.co.id/blog/jquery-adalah/>

Muhardian, A. 2017.Apa itu Git dan Kenapa Penting bagi Programmer? Retrieved from Petanikode: <https://www.petanikode.com/git-untuk-pemula/>

Waslaluddin. 2010. Internet dan Jaringan. Materi TIK, 105.

Wongso, V. O. 2017. Pengenalan Framework PHP. Retrieved from School of Information System: <https://sis.binus.ac.id/2017/12/18/pengenalan-framework-php>