

BAB 4

HASIL PEKERJAAN

4.1 Tahapan Implementasi

Pada pengerjaan proyek SIPPM di I2C Studio, penulis dan tim bersama-sama mengerjakan sesuai yang sudah ditugaskan melalui Trello. Terdapat beberapa modul pada proyek SIPPM ini, di antaranya terdapat modul penelitian, pengabdian masyarakat, pembuatan buku, dan hasil karya ilmiah. Penulis mendapat beberapa tugas yang mencakup semua modul tersebut. Namun penulis ingin fokus pada tugas yang berkaitan dengan modul penelitian karena dari tugas yang diberikan kepada penulis lebih banyak di modul penelitian. Pada modul penelitian ini, penulis mengerjakan bagian master data, yaitu membuat migrasi dan model untuk modul penelitian serta membuat *create, read, delete* (CRD) untuk bagian Prasyarat Program Penelitian. Bahasa pemrograman yang dipakai penulis adalah PHP dengan menggunakan *framework* Laravel serta Visual Studio Code sebagai kode editor. Selain itu, penulis juga menggunakan Github Desktop agar dapat melakukan *push* dan *pull* kode program yang telah dikerjakan oleh penulis dan rekan di I2C Studio. Untuk *database*, penulis menggunakan PostgreSQL. Sebelumnya, penulis perlu untuk melakukan instalasi PHP, Composer, NodeJS, dan PostgreSQL. Sebelum mengerjakan *master* data, penulis terlebih dahulu mempelajari penggunaan PHP dan Laravel serta mengikuti tes dari pihak I2C Studio

4.1.1 Instalasi PHP

Agar dapat menggunakan bahasa pemrograman PHP, maka perlu melakukan instalasi PHP. PHP ini dapat diunduh melalui laman resmi. *File zip* yang sudah diunduh dapat segera diekstrak. Kemudian menambahkan *path* yang telah diekstrak ke dalam *environment*. Untuk melihat versi PHP yang sudah ter-*install* pada perangkat dapat dilakukan mengetik perintah pada Command Prompt.

```
C:\Users\Lenovo>php -v
PHP 8.2.2 (cli) (built: Jan 31 2023 21:19:11) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.2, Copyright (c) Zend Technologies

C:\Users\Lenovo>
```

Gambar 4.1. 1 Perintah Command Prompt untuk melihat versi PHP

Gambar 4.1.1 adalah cara untuk melihat versi PHP yang sudah di-*install*. Caranya yaitu dengan mengetik perintah “`php -v`” pada Command Prompt. Setelah itu akan muncul versi PHP yang sudah di-*install*.

4.1.2 Instalasi Composer

Agar dapat menggunakan bahasa pemrograman PHP dan *framework* Laravel, maka diperlukan Composer. Composer dapat diunduh dengan mengunduh *file setup composer* dari laman resmi. Lalu *file setup* tersebut dijalankan, setelah itu pilih pilihan “*install for all users*”, kemudian tekan “*Next*” hingga akhir. Untuk melihat versi Composer yang sudah ter-*install* pada perangkat dapat dilakukan mengetik perintah pada Command Prompt.

```
C:\Users\Lenovo>composer

Composer version 2.5.3 2023-02-10 13:23:52

Usage:
  command [options] [arguments]

Options:
  -h, --help          Display help for the given command. When no command is given display help for the list command
  -q, --quiet         Do not output anything
  -V, --version       Display this application version
  --ansi[=no-ansi]    Force (or disable --no-ansi) ANSI output
  -n, --no-interaction Do not ask any interactive question
  --profile          Display timing and memory usage information
  --no-plugins        Never load plugins
  --no-scripts        Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=DIR  If specified, use the given directory as working directory.
  --no-cache          Prevent use of the cache
  -vvv[=vvv]           Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

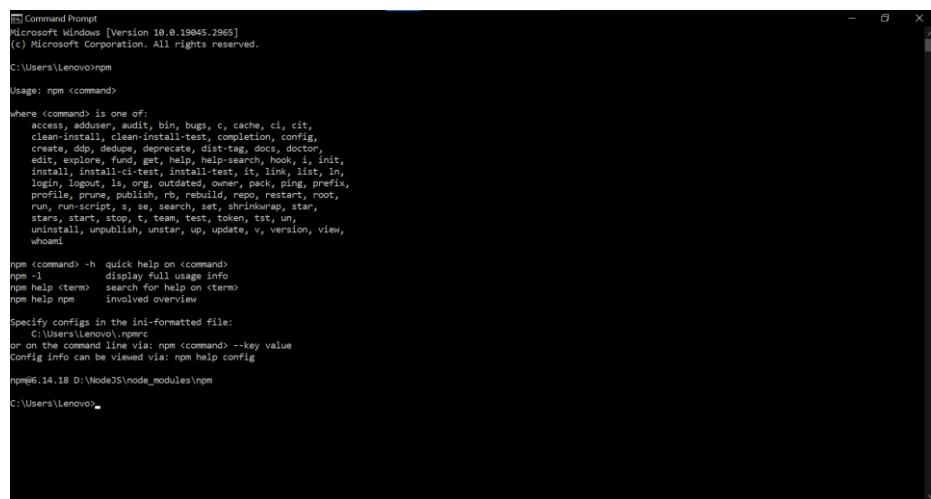
Available commands:
  about      Shows a short information about Composer
  archive   Creates an archive of this composer package
  audit     Checks if dependency versions are up-to-date for installed packages
  browser   [Open] Opens the package's repository URL or homepage in your browser
  bump      Increases the lower limit of your composer.json requirements to the currently installed versions
  check-platform-reqs Check that platform requirements are satisfied
  clear-cache [clearcache] Clears composer's internal package cache
  config     Sets config options
  create-project Creates new project from a package into given directory
  depends   [why] Shows which packages cause the given package to be installed
  diagnose  Diagnoses the system to identify common errors
```

Gambar 4.1. 2 Perintah Command Prompt untuk melihat versi Composer

Gambar 4.1.2 adalah cara untuk melihat versi Composer yang sudah di-*install*. Caranya adalah dengan mengetik perintah “`composer`” pada Command Prompt. Setelah itu akan muncul versi Composer yang sudah di-*install*.

4.1.3 Instalasi Node.js

CoreUI memerlukan adanya *dependency* dari NodeJS, oleh karena itu perlu melakukan instalasi Node.js. Node.js dapat diunduh dengan mengunduh file *setup* composer dari laman resmi. Lalu *file setup* tersebut dijalankan, kemudian tekan “*Next*” hingga akhir. Untuk melihat versi Node.js yang sudah ter-*install* pada perangkat dapat dilakukan mengetik perintah pada Command Prompt.



```
Command Prompt
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All Rights Reserved.

C:\Users\Lenovo>npm
Usage: npm <command>

where <command> is one of:
  access, adduser, audit, bin, bugs, c, cache, ci, cit,
  clean-install, clean-install-test, completion, config,
  create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
  edit, explore, fund, get, help, help-search, hook, i, init,
  install, install-ci-test, install-test, it, link, list, ln,
  ls, package, publish, profile, profiles, prefix,
  profile, prune, publish, rb, rebuild, repo, restart, root,
  run, run-script, s, se, search, set, shrinkwrap, star,
  stars, start, stop, t, team, test, token, tst, un,
  uninstall, unpublish, unstar, up, update, v, version, view,
  whoami

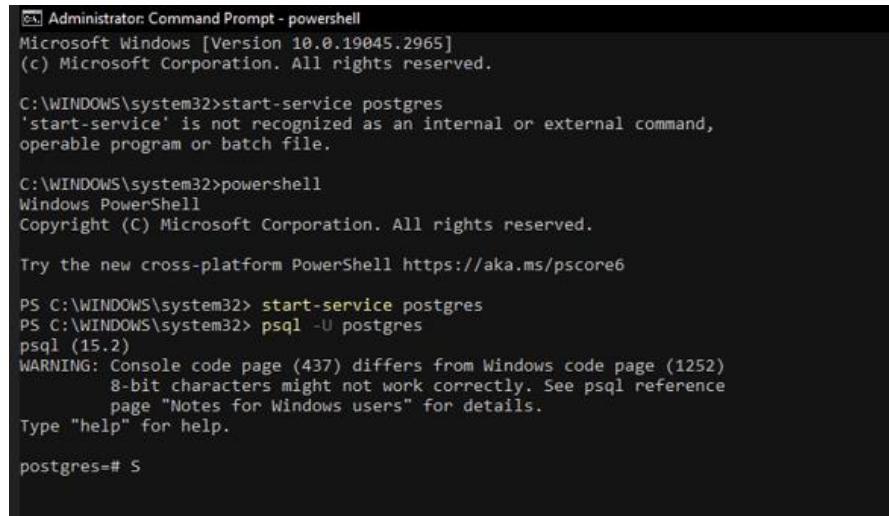
npm <command> -h    quick help on <command>
npm -l    display full usage info
npm help <term>  search for help on <term>
npm help npm  involved overview
Specify configs in the ini-formatted file:
  C:\Users\Lenovo\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config
npm@6.14.18 D:\NodeJS\node_modules\npm
C:\Users\Lenovo>
```

Gambar 4.1.3 Perintah Command Prompt untuk melihat versi Node.js

Gambar 4.1.3 adalah cara untuk melihat versi Node.js yang sudah di-*install*. Caranya adalah dengan mengetik perintah “*npm*” pada Command Prompt. Setelah itu akan muncul versi NodeJS yang sudah di-*install*.

4.1.4 Instalasi dan Membuat *Database* pada PostgreSQL

PostgreSQL dapat diunduh dengan mengunduh *file setup* dari laman resmi. Kemudian membuka file setup tersebut dan tekan “*Next*” hingga halaman kata sandi. Masukkan kata sandi yang diinginkan. Untuk *port server* menggunakan *default* atau tidak perlu diubah-ubah. Untuk melihat versi PostgreSQL yang sudah ter-*install* pada perangkat dapat dilakukan mengetik perintah pada Command Prompt.



```

Administrator: Command Prompt - powershell
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>start-service postgres
'start-service' is not recognized as an internal or external command,
operable program or batch file.

C:\WINDOWS\system32>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> start-service postgres
PS C:\WINDOWS\system32> psql -U postgres
psql (15.2)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

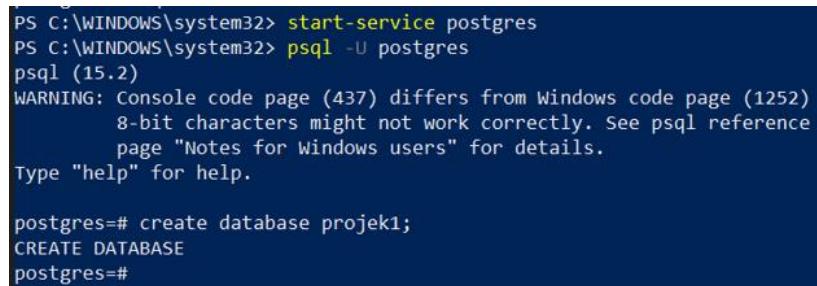
postgres=# S

```

Gambar 4.1. 4 Perintah Command Prompt untuk melihat versi PostgreSQL

Gambar 4.1.4 adalah cara untuk melihat versi PostgreSQL yang sudah di-*install*. Caranya adalah dengan mengetik perintah “powershell” pada Command Prompt, lalu tekan enter, kemudian ketik “start-service postgres” lalu enter dan ketik “psql -U postgres”. Lalu akan muncul versi PostgreSQL yang sudah di-*install*.

Untuk membuat database pada PostgreSQL dapat mengetik perintah pada Windows Powershell.



```

PS C:\WINDOWS\system32> start-service postgres
PS C:\WINDOWS\system32> psql -U postgres
psql (15.2)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# create database projek1;
CREATE DATABASE
postgres=#

```

Gambar 4.1. 5 Perintah Powershell membuat database baru

Gambar 4.1.5 adalah perintah yang penulis ketik pada Windows Powershell untuk membuat database baru pada PostgreSQL, yaitu dengan cara mengetik perintah “create database” dilanjutkan dengan nama database.

4.1.5 Instalasi Laravel Collective

Agar dapat menggunakan Laravel Collective, maka perlu dilakukan instalasi terlebih dahulu.

```
D:\I2C\tugas1juan>composer require laravelcollective/html
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been updated
Running composer update laravelcollective/html
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
- Upgrading laravelcollective/html (v6.4.0 => v6.4.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
- Downloading laravelcollective/html (v6.4.1)
- Upgrading laravelcollective/html (v6.4.0 => v6.4.1): Extracting archive
```

Gambar 4.1.6 Perintah Command Prompt *install* Laravel Collective

Gambar 4.1.6 adalah cara untuk meng-install Laravel Collective, yaitu dengan mengetik perintah “composer require laravelcollective/html” pada Command Prompt. Lalu menunggu hingga Laravel Collective siap digunakan.

4.1.6 Membuat projek baru pada Laravel

Pada Laravel, untuk membuat projek baru dapat dilakukan dengan mengetik perintah pada Command Prompt.

```
D:\I2C>composer create-project --prefer-dist laravel/laravel projek1
Creating a "laravel/laravel" project at "./projek1"
Installing laravel/laravel (v10.2.2)
- Downloading laravel/laravel (v10.2.2)
- Installing laravel/laravel (v10.2.2): Extracting archive
Created project in D:\I2C\projek1
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
```

Gambar 4.1.7 Perintah Command Prompt membuat projek baru

Gambar 4.1.7 adalah cara untuk membuat projek baru pada Laravel, yaitu dengan mengetik perintah “create-project –prefer-dist laravel/laravel” lalu dilanjutkan dengan nama projek pada Command Prompt. Lalu menunggu hingga projek baru siap digunakan.

4.1.7 Membuat Migrasi dan Model dengan Laravel

Laravel memudahkan penulis dalam membuat migrasi dan model. Untuk membuat migrasi dan model, dapat dilakukan dengan mengetik perintah pada Command Prompt.

```
D:\I2C\tugas1juan>php artisan make:migration a
[INFO] Migration [D:\I2C\tugas1juan\database\migrations/2023_05_27_084747_a.php] created successfully.

D:\I2C\tugas1juan>php artisan make:model a
[INFO] Model [D:\I2C\tugas1juan\app\Models/a.php] created successfully.

D:\I2C\tugas1juan>
```

Gambar 4.1.8 Perintah Command Prompt migrasi dan model

Gambar 4.1.8 adalah cara untuk membuat migrasi dan model pada Laravel. Untuk membuat migrasi, caranya adalah dengan mengetik “php artisan make:migration” dilanjutkan dengan nama dari migrasi tersebut pada Command Prompt. Untuk membuat model, caranya adalah dengan mengetik “php artisan make:model” dilanjutkan dengan nama dari model tersebut pada Command Prompt.

4.1.8 Melakukan Seeding

Tabel yang baru dibuat tentunya masih kosong atau belum memiliki data. Oleh karena itu, perlu memasukkan data *dummy* ke dalam tabel kosong tersebut. Caranya adalah dengan memanfaatkan fitur dari Laravel yaitu dengan menggunakan seeder. Setelah membuat kode program pada seeder, selanjutnya perlu dilakukan seeding agar data dummy yang telah dibuat dapat masuk ke dalam tabel. Caranya adalah mengetik perintah pada Command Prompt.

```
Windows PowerShell
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

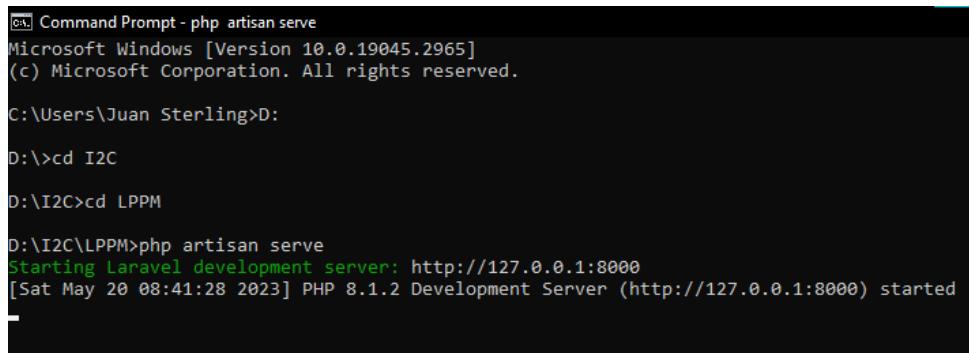
C:\Users\Lenovo>D:
D:>cd I2C
D:\I2C>cd LPPM
D:\I2C\LPPM>php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
```

Gambar 4.1.9 Perintah Command Prompt seeding

Gambar 4.1.9 adalah cara untuk melakukan seeding, yaitu dengan cara mengetik perintah “php artisan migrate:fresh --seed” pada Command Prompt.

4.1.9 Mengakses halaman web

Untuk dapat mengakses halaman web, penulis perlu mengetik perintah pada Command Prompt untuk menjalankan PHP dan Laravel.



```
cmd Command Prompt - php artisan serve
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Juan Sterling>D:
D:\>cd I2C
D:\I2C>cd LPPM
D:\I2C\LPPM>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sat May 20 08:41:28 2023] PHP 8.1.2 Development Server (http://127.0.0.1:8000) started
```

Gambar 4.1. 10 Perintah Command Prompt menjalankan PHP dan Laravel

Gambar 4.1.10 adalah perintah pada Command Prompt untuk menjalankan PHP dan Laravel. Kemudian menyalin <http://127.0.0.1:8000> pada *web browser* untuk mengakses halaman web.

4.2 Membuat Projek Sederhana

Pertama-tama penulis mempelajari bahasa pemrograman PHP dan framework Laravel. Penulis berlatih dengan membuat proyek sederhana. Dalam projek ini, penulis membuat tabel untuk mahasiswa, mata kuliah, dan semester. Ketiga tabel tersebut nantinya akan saling berelasi. Dalam projek ini penulis mempelajari cara membuat dan menggunakan *master layout*, *login*, *sign up*, migrasi, model, *read*, *create*, *update*, *delete* dengan PHP dan Laravel. Pada projek ini juga penulis belajar menggunakan PostgreSQL. Selain itu juga mempelajari fitur-fitur lain seperti DataTables, Bootstrap, SWAL, Select2. Hal pertama yang dilakukan setelah membuat projek baru adalah membuat *database* baru pada PostgreSQL. Setelah itu menghubungkan PostgreSQL dengan projek sederhana ini.

```

11 DB_CONNECTION=pgsql
12 DB_HOST=127.0.0.1
13 DB_PORT=5432
14 DB_DATABASE=tes3
15 DB_USERNAME=postgres
16 DB_PASSWORD=

```

Gambar 4.2. 1 Kode program .env projek sederhana

Gambar 4.2.1 adalah kode program pada file .env untuk menghubungkan PostgreSQL dengan projek sederhana ini. Nama database yang penulis buat pada PostgreSQL adalah “tes3”. Username yang digunakan adalah “postgres” sesuai dengan username pada PostgreSQL.

4.2.1 Master Layout

Selanjutnya adalah membuat *master* untuk *layout*. Ini dilakukan untuk membuat master dari tampilan web. Dalam master ini, akan terdapat kode program untuk *import/install* fitur-fitur yang nantinya akan digunakan.

```

1 <!DOCTYPE html>
2 <html lang="en" data-bs-theme="light">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/v/rt/bs-5.1.3/r5/dt-1.13.2/datatables.min.css" />
8   <script type="text/javascript" src="https://cdn.datatables.net/v/rt/bs-5.1.3/r5/dt-1.13.2/datatables.min.js" />
9
10  <!-- Database -->
11  <script type="text/javascript" src="https://cdn.datatables.net/v/rt/bs-5.1.3/r5/dt-1.13.2/datatables.min.js" />
12  <script type="text/javascript" src="https://cdn.datatables.net/v/rt/bs-5.1.3/r5/dt-1.13.2/datatables.min.js" />
13
14  <!-- Bootstrap -->
15  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
16    integrity="sha384-GLhlTq6OwSI6rZL2cBZ0l5XQjJnK2t7Q1vvhVmCk5H2yU7PwYl4qYKzZD8hJ0" crossorigin="anonymous">
17  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4ZUI0q6E2d+uq98ZxqKkO+o16E2u+8vWQ8F7JIXtp43Mz9f/JN3" crossorigin="anonymous">
18  </script>
19  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" />
20  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/umd/bootstrap.min.js" integrity="sha384-O8tR+u5SvCwvKvSAtADfZJdIw07tgbHkmTOIVdA238vF20K4SRZ5+aqKgkf/x" crossorigin="anonymous">
21  </script>
22  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-GLhlTq6OwSI6rZL2cBZ0l5XQjJnK2t7Q1vvhVmCk5H2yU7PwYl4qYKzZD8hJ0" crossorigin="anonymous">
23  </script>
24
25  <link href="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.1/dist/semantic.min.css" rel="stylesheet" />
26  <script src="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.1/dist/semantic.min.js" />
27  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/select2-bootstrap-theme/4.0.3/select2-bootstrap.min.css" integrity="sha5d14c3f1w8udM1sp5YK9wU1Vb730Gcp0mIAeyRqdz93jpsf4dwia|110e--" crossorigin="anonymous" refererpoli
28  <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0/dist/js/select2.min.js" />
29  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
30  <title>Document</title>
31  </head>
32
33  <body>
34
35  <nav class="navbar navbar-expand-lg fixed-top bg-dark" data-bs-theme="dark">
36    <div class="container-fluid">
37
38      <a class="navbar-brand" href="/home/123" />
39      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
40        <span class="navbar-toggler-icon"></span>
41      </button>
42      <div class="collapse navbar-collapse" id="navbarSupportedContent">
43        <ul class="navbar-nav me-auto mb-2 d-lg-block">
44          <li class="nav-item">
45            <a class="nav-link active" aria-current="page" href="/home">Home</a>
46          </li>
47          <li class="nav-item">
48            <a class="nav-link" href="#">Semester</a>
49          </li>
50          <li class="nav-item">
51            <a class="nav-link" href="#">Khalid</a>
52          </li>
53          <li class="nav-item dropdown">
54            <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
55              DropDown
56            </a>
57            <ul class="dropdown-menu">
58              <li><a href="#">Action</a></li>
59              <li><a href="#">Another Action</a></li>
60              <li><a href="#">Something else here</a></li>
61            </ul>
62          </li>
63        </ul>
64      </div>
65    </div>
66  </nav>

```

Gambar 4.2. 2 Kode program *master layout* projek sederhana 1



```

1 <head>
2   <title>Sederhana</title>
3   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
4   <link href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.2.1/css/all.min.css" rel="stylesheet">
5   <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
6   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJhffZOUEhtlyfD2Hcv7JreHnUaa0lEnoHnqZqrKM2Kew1Fw�oE+R7xgqD+S" crossorigin="anonymous"></script>
7   <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-64m06t4d9tX8ZtC+oXq50o7TQ9v4HnqK2u8yKkKjFQ==" crossorigin="anonymous"></script>
8   <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/fontawesome-all.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
9   <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/webfontloader.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
10  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/voyager.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
11  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/icomoon.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
12  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/flagkit.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
13  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/brands.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
14  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/fontawesome.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
15  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/voyager.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
16  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/icomoon.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
17  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/flagkit.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
18  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/brands.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
19  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/fontawesome.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
20  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/voyager.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
21  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/icomoon.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
22  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/flagkit.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
23  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/brands.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
24  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/fontawesome.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
25  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/voyager.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
26  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/icomoon.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
27  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/flagkit.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
28  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/brands.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
29  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/fontawesome.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">
30  <script src="https://cdn.jsdelivr.net/npm/@fontawesome/fontawesome-free@6.2.1/js/voyager.min.js" integrity="sha384-7Zkq0ZIy+jvJLWj9DyOQn0n4K4G0HwzQc2eOYQ5LJdPZV1YB&ts=6212438d">

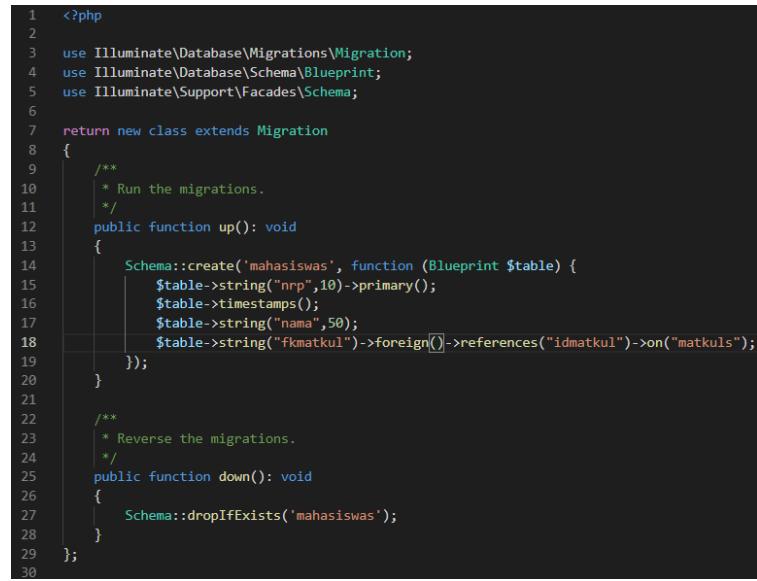
```

Gambar 4.2. 3 Kode program *master layout* projek sederhana 2

Gambar 4.2.2 dan gambar 4.2.3 adalah kode program yang telah penulis kerjakan untuk membuat *master layout* pada projek sederhana. Pada kode program tersebut terdapat kode program untuk melakukan *import/install* CDN terhadap fitur-fitur atau *library* yang akan digunakan, seperti Bootstraps, DataTables, SWAL, dan Select2. Terdapat juga kode program untuk membuat *navigation bar*. Untuk dapat menggunakan master layout ini, maka pada tampilan atau view perlu menambahkan kode program “*extends(“master”)*” dan “*section(“content”)*”.

4.2.2 Migrasi dan Model

Selanjutnya penulis membuat migrasi dan model untuk tabel mahasiswa, mata kuliah, dan semester.



```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('mahasiswa', function (Blueprint $table) {
15             $table->string('nrp',10)->primary();
16             $table->timestamps();
17             $table->string("nama",50);
18             $table->string("fkmatkul")->foreign()>references("idmatkul")>on("matkuls");
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists('mahasiswa');
28     }
29 };
30

```

Gambar 4.2. 4 kode program migrasi mahasiswa

Gambar 4.2.4 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel mahasiswa. Terdapat kolom nrp, nama, dan fkmatkul. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Database\Factories\mahasiswafactoryFactory;
8
9  class mahasiswa extends Model
10 {
11
12     use HasFactory;
13     protected $primaryKey="nrp";
14     public $incrementing = false;
15
16     protected static function newFactory(){
17         return mahasiswafactoryFactory::new();
18     }
19     public function getMatkul(){
20         return $this->hasMany(matkul::class,"idmatkul","fkmatkul");
21     }
22 }
23

```

Gambar 4.2. 5 kode program model mahasiswa

Gambar 4.2.5 adalah kode program yang telah penulis kerjakan untuk membuat model untuk tabel mahasiswa. Terdapat juga kode program untuk membuat relasi dengan tabel matkuls. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut.

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('matkuls', function (Blueprint $table) {
16             $table->string("idmatkul")->primary();
17             $table->timestamps();
18             $table->string("matkul");
19             $table->string("skls");
20             $table->bigInteger("fksemester")->foreign()->references("idsemester")->on("semesters");
21         });
22     }
23
24     /**
25      * Reverse the migrations.
26      */
27     public function down(): void
28     {
29         Schema::dropIfExists('matkuls');
30     }
31 };
32

```

Gambar 4.2. 6 kode program migrasi mata kuliah

Gambar 4.2.6 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel matkuls. Terdapat kolom idmatkul, matkul, sks, dan fksemester. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut.

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class matkul extends Model
9 {
10     use HasFactory;
11     protected $primaryKey = "idmatkul";
12     public $incrementing = false;
13
14     public function getSemester(){
15         return $this->belongsTo(semester::class, 'fksemester');
16     }
17 }
```

Gambar 4.2. 7 kode program model mata kuliah

Gambar 4.2.7 adalah kode program yang telah penulis kerjakan untuk membuat model untuk tabel matkuls. Terdapat juga kode program untuk membuat relasi dengan tabel semesters.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('semesters', function (Blueprint $table) {
15             $table->string("idsemester")->primary();
16             $table->timestamps();
17             $table->string("semester");
18         });
19     }
20
21     /**
22      * Reverse the migrations.
23      */
24     public function down(): void
25     {
26         Schema::dropIfExists('semesters');
27     }
28 };
```

Gambar 4.2. 8 kode program migrasi semester

Gambar 4.2.8 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel semesters. Terdapat kolom idsemester dan semester.

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class semester extends Model
9 {
10     protected $primaryKey="idsemester";
11     public $incrementing = false;
12     use HasFactory;
13 }

```

Gambar 4.2. 9 kode program model semester

Gambar 4.2.9 adalah kode program yang telah penulis kerjakan untuk membuat model untuk tabel semesters.

4.2.3 *Read*

Selanjutnya penulis mengerjakan untuk melakukan read. Penulis terlebih dahulu membuat route untuk menampilkan semua data dari tabel mahasiswa, mata kuliah, dan semester.

```

28 |     Route::get('/home', [MahasiswaController::class, "index"]);
29 |     Route::get('/matkul', [MatkulController::class, "index"]);
30 |     Route::get('/semester', [SemesterController::class, "index"]);

```

Gambar 4.2. 10 kode program *route read* projek sederhana

Gambar 4.2.10 adalah kode program yang penulis kerjakan untuk membuat *route read* untuk menampilkan data mahasiswa, mata kuliah, dan semester. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *get*, karena berfungsi untuk menampilkan data. Untuk *home* akan menampilkan data mahasiswa, matkul untuk menampilkan data mata kuliah, dan semester untuk menampilkan data semester. Tiap tabel memiliki *controller* masing-masing. Untuk menampilkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “index”.

```

17 <table id=table_id class="display">
18   <thead>
19     <th>NRP</th>
20     <th>>Nama</th>
21     <th>Mata Kuliah</th>
22     <th>Photo</th>
23     <th>action</th>
24   </thead>
25   <tbody>
26     @foreach ($mahasiswa as $item)
27       <tr>
28         <td>
29           {{ $item->nrp }}
30         </td>
31         <td>
32           {{ $item->nama }}
33         </td>
34         <td>
35           {{ $item->getMatkul[0]->matkul }}
36         </td>
37         <td>
38           
39         </td>
40         <td>
41           <button class="btn btn-danger" onclick="del('{{ $item->nrp }}')>Delete</button>
42           <a href="/home/edit/{{ $item->nrp }}><button class="btn btn-warning">Edit</button></a>
43         </td>
44       </tr>
45     @endforeach
46   </tbody>
47 </table>

```

Gambar 4.2. 11 kode program tampilan mahasiswa

Gambar 4.2.11 adalah kode program yang telah penulis kerjakan untuk membuat tampilan mahasiswa. Pada tampilan ini akan menampilkan tabel yang menggunakan DataTables. Tabel ini akan menampilkan data-data mahasiswa. Terdapat juga tombol *edit* dan *delete*.

```

19 <table id=table_id class="display">
20   <thead>
21     <th>ID Mata Kuliah</th>
22     <th>Mata Kuliah</th>
23     <th>Total SKS</th>
24     <th>Semester</th>
25     <th>action</th>
26   </thead>
27   <tbody>
28     @foreach ($matkul as $item)
29       <tr>
30         <td>
31           {{ $item->idmatkul }}
32         </td>
33         <td>
34           {{ $item->matkul }}
35         </td>
36         <td>
37           {{ $item->skls }}
38         </td>
39         <td>
40           {{ $item->fksemester }}
41         </td>
42         <td>
43           <button class="btn btn-danger" onclick="del('{{ $item->idmatkul }}')>Delete</button>
44           <a href="/matkul/edit/{{ $item->idmatkul }}><button class="btn btn-warning">Edit</button></a>
45         </td>
46       </tr>
47     @endforeach
48   </tbody>
49 </table>
50 <script>
51   $(document).ready(function() {
52     $('#table_id').DataTable();
53   });
54

```

Gambar 4.2. 12 kode program tampilan mata kuliah

Gambar 4.2.12 adalah kode program yang telah penulis kerjakan untuk membuat tampilan mata kuliah. Pada tampilan ini akan menampilkan tabel yang

menggunakan DataTables. Tabel ini akan menampilkan data-data mata kuliah. Terdapat juga tombol *edit* dan *delete*.

```

16 |     <table id=table_id class="display">
17 |       <thead>
18 |         <th>ID</th>
19 |         <th>Semester</th>
20 |         <th>action</th>
21 |     </thead>
22 |     <tbody>
23 |       @foreach ($semester as $item)
24 |         <tr>
25 |           <td>
26 |             {{ $item->idsemester }}
27 |           </td>
28 |           <td>
29 |             {{ $item->semester }}
30 |           </td>
31 |
32 |           <td>
33 |             <button class="btn btn-danger" onclick="del('{{ $item->idsemester }}')>Delete</button>
34 |             <a href="/semester/edit/{{ $item->idsemester }}><button class="btn btn-warning">Edit</button></a>
35 |           </td>
36 |         </tr>
37 |       @endforeach
38 |     </tbody>
39 |   </table>
40 |   <script>
41 |     $(document).ready(function() {
42 |       $('#table_id').DataTable();
43 |     });

```

Gambar 4.2. 13 kode program tampilan semester

Gambar 4.2.13 adalah kode program yang telah penulis kerjakan untuk membuat tampilan semester. Pada tampilan ini akan menampilkan tabel yang menggunakan DataTables. Tabel ini akan menampilkan data-data semester. Terdapat juga tombol *edit* dan *delete*.

```

17 | public function index(Request $request)
18 | {
19 |   if($request->delcom == 1){
20 |     $delmahasiswa = mahasiswa::firstWhere("nrp",$request->nrp);
21 |     $delmahasiswa->delete();
22 |   }
23 |
24 |   $mahasiswas = mahasiswa::all();
25 |   $matkuls = matkul::all();
26 |   $matkulcoll = [];
27 |   foreach($matkuls as $item){
28 |     $matkulcoll[$item->idmatkul] = "Mata Kuliah ". $item->matkul;
29 |   }
30 |   // dd($matkulcoll);
31 |   // return view("mahasiswa.welcome",["mahasiswa"=>$mahasiswas,"matkul"=>$matkuls]);
32 |   return view("mahasiswa.welcome",["mahasiswa"=>$mahasiswas,"matkul"=>$matkulcoll]);
33 |
34 | }

```

Gambar 4.2. 14 kode program controller read mahasiswa

Gambar 4.2.14 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data mahasiswa.

```

18     public function index(Request $request)
19     {
20
21         if($request->delcom == 1){
22             $delmatkul= matkul::firstWhere("idmatkul",$request->idmatkul);
23             $delmatkul->delete();
24         }
25
26         $matkuls = matkul::all();
27         $semesters = semester::all();
28         $semestercoll = [];
29         foreach($semesters as $item){
30             $semestercoll[$item->idsemester] = "Semester ".$item->semester;
31         }
32
33         // return view("matkul.matkul",["matkul"=>$matkuls,"semester"=>$semesters]);
34         return view("matkul.matkul",["matkul"=>$matkuls,"semester"=>$semestercoll]);
35     }
36

```

Gambar 4.2. 15 kode program *controller read matkul*

Gambar 4.2.15 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data mata kuliah.

```

16     public function index(Request $request)
17     {
18         if($request->delcom == 1){
19             $delsemester= semester::firstWhere("idsemester",$request->idsemester);
20             $delsemester->delete();
21         }
22
23         $semesters = semester::all();
24         return view("semester.semester",["semester"=>$semesters]);
25     }
26

```

Gambar 4.2. 16 kode program *controller read semester*

Gambar 4.2.16 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data semester.

Data pada tabel mahasiswa, mata kuliah, dan semester masih kosong. Untuk memastikan bahwa kode program yang telah penulis kerjakan dapat menampilkan data dengan baik, maka tabel-tabel kosong tersebut perlu diisi data *dummy*. Untuk mengisi data *dummy*, penulis menggunakan fitur *factory* dan *seeder* dari Laravel.

```

1  <?php
2
3  namespace Database\Factories;
4  use App\Models\mahasiswa;
5
6  use Illuminate\Database\Eloquent\Factories\Factory;
7
8  /**
9  * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Model>
10 */
11 class mahasiswaFactory extends Factory
12 {
13     protected $model= mahasiswa::class;
14
15     /**
16      * Define the model's default state.
17      *
18      * @return array<string, mixed>
19     */
20     public function definition(): array
21     {
22         return [
23             "nrp"=>"2072". strval(fake()->unique()->numerify("###")),
24             "nama"=> fake()->name(),
25             "fkmatkul"=>"IN20". strval(fake()->numberBetween(101, 105))
26         ];
27     }
28

```

Gambar 4.2. 17 kode program *factory mahasiswa*

Gambar 4.2.17 adalah kode program untuk membuat *factory* untuk membuat data *dummy* pada tabel mahasiswa. Kolom yang dibuat data *dummy* adalah kolom *nrp*, *nama*, dan *mata kuliah*.

```

1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use App\Models\mahasiswa;
8  class mahasiswaSeeder extends Seeder
9  {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         $mahasiswa = mahasiswa::factory()->count(20)->make();
16
17         foreach($mahasiswa as $item){
18             $item->save();
19         }
20     }
21 }
22
23

```

Gambar 4.2. 18 kode program *seeder* mahasiswa

Gambar 4.2.18 adalah kode program untuk membuat *seeder* untuk mengisi data *dummy* pada tabel mahasiswa untuk kolom *nrp*, *nama*, dan *mata kuliah*.

NRP	Nama	Mata Kuliah	Photo	action
2072016	Dr. Santos Huel PhD	Allene Turner		<button>Delete</button> <button>Edit</button>
2072043	Samara Lockman	Raphael Green		<button>Delete</button> <button>Edit</button>
2072094	Victor Hoppe	Raphael Green		<button>Delete</button> <button>Edit</button>
2072100	Percy Harber	Jordan Romaguera		<button>Delete</button> <button>Edit</button>
2072137	Mr. Abdullah Schmeler	Keyshawn Russel		<button>Delete</button> <button>Edit</button>
2072150	Haskell Hodkiewicz	Allene Turner		<button>Delete</button> <button>Edit</button>

Gambar 4.2. 19 tampilan mahasiswa

Gambar 4.2.19 adalah tampilan untuk mahasiswa. Terdapat tabel yang menampilkan isi dari data mahasiswa. Terdapat juga bagian untuk menambahkan data mahasiswa serta tombol *edit* dan *delete*.

```

1 <?php
2
3 namespace Database\Factories;
4
5 use Illuminate\Database\Eloquent\Factories\Factory;
6
7 /**
8 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\matkul>
9 */
10 class matkulFactory extends Factory
{
11     /**
12      * Define the model's default state.
13      *
14      * @return array<string, mixed>
15      */
16     public function definition(): array
17     {
18
19         return [
20             "idmatkul"=>"IN20". strval(fake()->unique()->numberBetween([101, 105])),
21             "matkul"=> fake()->name(),
22             "sks"=>fake()->numberBetween(1,9),
23             "fksemester"=>fake()->unique()->numberBetween(1,10)
24             //
25         ];
26     }
27 }
28
29

```

Gambar 4.2. 20 kode program *factory* mata kuliah

Gambar 4.2.20 adalah kode program untuk membuat *factory* untuk membuat data *dummy* pada tabel mata kuliah. Kolom yang dibuat data *dummy* adalah kolom id mata kuliah, mata kuliah, sks, dan semester.

```

1 <?php
2
3 namespace Database\Seeders;
4
5 use App\Models\matkul;
6 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7 use Illuminate\Database\Seeder;
8
9 class matkulseeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $matkul = matkul::factory()->count([5])->make();
17
18         foreach($matkul as $item){
19             $item->save();
20         }
21     }
22 }
23

```

Gambar 4.2. 21 kode program *seeder* mata kuliah

Gambar 4.2.21 adalah kode program yang telah penulis kerjakan untuk membuat *seeder* untuk mengisi data *dummy* pada tabel mata kuliah untuk id mata kuliah, mata kuliah, sks, dan semester.

ID Mata Kuliah	Mata Kuliah	Total SKS	Semester	action
IN20101	Bradly Gorczany	1	9	<button>Delete</button> <button>Edit</button>
IN20102	Chester O'Conner	1	7	<button>Delete</button> <button>Edit</button>
IN20103	Guadalupe Lubowitz PhD	8	2	<button>Delete</button> <button>Edit</button>
IN20104	Veda Tromp	5	8	<button>Delete</button> <button>Edit</button>
IN20105	Agustina Cummerata Jr.	2	4	<button>Delete</button> <button>Edit</button>

Gambar 4.2.22 tampilan mata kuliah

Gambar 4.2.22 adalah tampilan untuk mata kuliah. Terdapat tabel yang menampilkan isi dari data mata kuliah. Terdapat juga bagian untuk menambahkan data mata kuliah serta tombol *edit* dan *delete*.

```

1 <?php
2
3 namespace Database\Factories;
4
5 use Illuminate\Database\Eloquent\Factories\Factory;
6
7 /**
8 * @extends \Illuminate\Database\Eloquent\Factories\Factory<App\Models\semester>
9 */
10 class semesterFactory extends Factory
{
11
12 /**
13 * Define the model's default state.
14 *
15 * @return array<string, mixed>
16 */
17 public function definition(): array
18 {
19     $sems = fake()->unique()->numberBetween([1,10]);
20     return [
21         "idsemester"=>$sems,
22         "Semester"=> $sems
23         // ...
24     ];
25 }
26
27 }
```

Gambar 4.2.23 kode program *factory* semester

Gambar 4.2.23 adalah kode program yang telah penulis kerjakan untuk membuat *factory* untuk membuat data *dummy* pada tabel semester. Kolom yang dibuat data *dummy* adalah kolom idsemester dan semester.

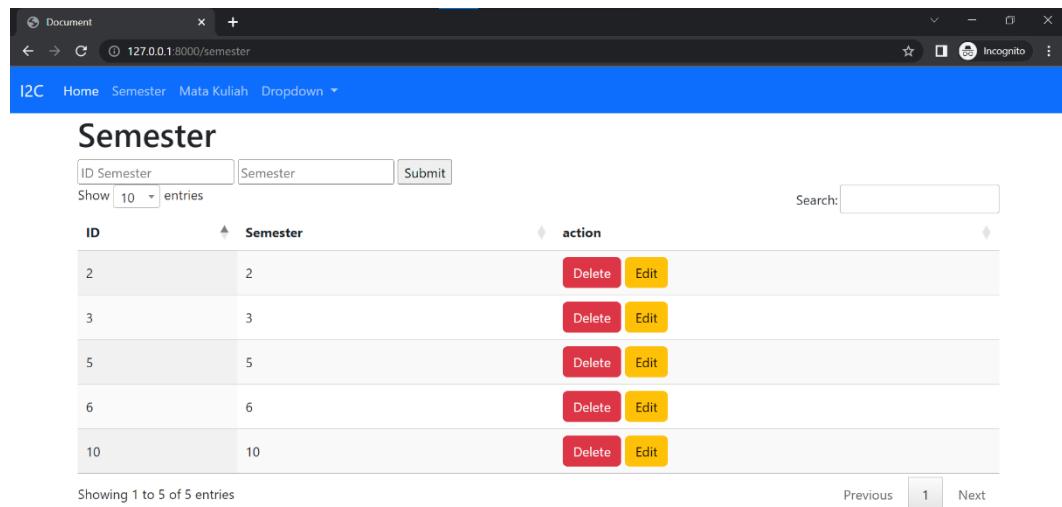
```

1 <?php
2
3 namespace Database\Seeders;
4
5 use App\Models\semester;
6 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7 use Illuminate\Database\Seeder;
8
9 class semesterSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $semester = semester::factory()->count(5)->make();
17
18         foreach($semester as $item){
19             $item->save();
20         }
21     }
22 }
23

```

Gambar 4.2. 24 kode program *seeder* semester

Gambar 4.2.24 adalah kode program yang penulis kerjakan untuk membuat *seeder* untuk mengisi data *dummy* pada tabel semester untuk idsemester dan semester.



The screenshot shows a web application interface for managing semesters. At the top, there's a navigation bar with links for 'Home', 'Semester', 'Mata Kuliah', and 'Dropdown'. Below the navigation, the title 'Semester' is displayed. A search bar labeled 'Search:' is present. On the left, there are input fields for 'ID Semester' and 'Semester', and a 'Submit' button. Below these, a table lists five entries:

ID	Semester	action
2	2	<button>Delete</button> <button>Edit</button>
3	3	<button>Delete</button> <button>Edit</button>
5	5	<button>Delete</button> <button>Edit</button>
6	6	<button>Delete</button> <button>Edit</button>
10	10	<button>Delete</button> <button>Edit</button>

At the bottom of the table, it says 'Showing 1 to 5 of 5 entries'. There are also 'Previous' and 'Next' buttons.

Gambar 4.2. 25 tampilan semester

Gambar 4.2.25 adalah tampilan untuk semester. Terdapat tabel yang menampilkan isi dari data semester. Terdapat juga bagian untuk menambahkan data semester serta tombol *edit* dan *delete*.

4.2.4 Create

Selanjutnya adalah membuat bagian untuk *create* atau menambahkan data. Penulis terlebih dahulu membuat *route* untuk menambah data untuk tabel mahasiswa, mata kuliah, dan semester.

```
36 |     Route::post('/add', [MahasiswaController::class, "store"]);
37 |     Route::post('/addmatkul', [MatkulController::class, "store"]);
38 |     Route::post('/addsemester', [SemesterController::class, "store"]);
```

Gambar 4.2. 26 kode program *route create* projek sederhana

Gambar 4.2.26 adalah kode program yang telah penulis kerjakan untuk membuat *route create* untuk menambahkan data mahasiswa, mata kuliah, dan semester. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *post*, karena berfungsi untuk menambahkan data. Untuk “add” akan menambahkan data mahasiswa, “addmatkul” akan menambahkan data mata kuliah, dan untuk “addsemester” untuk menambahkan data semester. Untuk menambahkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “store”

```
8 |     <h1>Mahasiswa</h1>
9 |     {{Form::open(["url"=>"/add", "files"=>"true"])}}
10|     {{Form::text("nrp",null,[{"placeholder"=>"nrp"}])}}
11|     {{Form::text("nama",null,[{"placeholder"=>"nama"}])}}
12|     {{Form::select("fkmatkul",$matkul)}}
13|     {{Form::file("photo",["accept"=>".jpg,.png","class"=>"form form-control"])}}
14|     {{Form::submit("submit")}}
15|     {{Form::close()}}
```

Gambar 4.2. 27 kode program tampilan tambah mahasiswa

Gambar 4.2.27 adalah kode program yang telah penulis kerjakan untuk membuat tampilan untuk menambahkan data mahasiswa. Terdapat *input text* untuk menambahkan data NRP dan Nama, kemudian terdapat *select* yang menggunakan fitur Select2, di mana *select* ini berisi mata kuliah, terdapat field untuk mengunggah gambar, dan terdapat tombol “submit”.

```
8 |     <h1>Mata Kuliah</h1>
9 |     {{Form::open(["url"=>"/addmatkul"])}}
10|     {{Form::text("idmatkul",null,[{"placeholder"=>"ID Mata Kuliah"}])}}
11|     {{Form::text("matkul",null,[{"placeholder"=>"Mata Kuliah"}])}}
12|     {{Form::select("skls",["2"=>"2 SKS","3"=>"3 SKS","4"=>"4 SKS"])}}
13|     {{Form::select("fksemester",$semester)}}
14|     <div>
15|         {{Form::submit("Submit")}}
16|     </div>
17|     {{Form::close()}}
```

Gambar 4.2. 28 kode program tampilan tambah mata kuliah

Gambar 4.2.28 adalah kode program yang penulis kerjakan membuat tampilan untuk menambahkan data mata kuliah. Tampilan ini hanya dapat muncul jika *user* memiliki *role* “admin”. Terdapat *input text* untuk menambahkan data ID Mata Kuliah dan Mata Kuliah, kemudian terdapat *select* yang menggunakan fitur Select2, di mana *select* ini berisi SKS, lalu terdapat *select* yang menggunakan fitur Select2, di mana *select* ini berisi Semester, dan terdapat tombol “submit”.

```

8   <h1>Semester</h1>
9   @if (Auth::user()->role == "admin")
10  {{Form::open(["url"=>"/addsemester"])}}
11  {{Form::text("idsemester",null,[{"placeholder"=>"ID Semester"}])}}
12  {{Form::text("semester",null,[{"placeholder"=>"Semester"}])}}
13  {{Form::submit("Submit")}}
14  {{Form::close()}}
15  @endif

```

Gambar 4.2. 29 kode program tampilan tambah semester

Gambar 4.2.29 adalah kode program yang penulis kerjakan membuat tampilan untuk menambahkan data semester. Tampilan ini hanya dapat muncul jika *user* memiliki *role* “admin”. Terdapat *input text* untuk menambahkan data ID Semester dan Semester, kemudian terdapat tombol “submit”.

```

47  public function store(Request $request)
48  {
49      $datanrp = $request->nrp;
50      $datanama = $request->nama;
51      $datafk = $request->fkmatkul;
52      $dataphoto = $request->file("photo");
53      $filename = $datanrp. $dataphoto->getClientOriginalname();
54      $destination = public_path('/image');
55      $dataphoto->move($destination,$filename);
56      $mahasiswa = new Mahasiswa();
57      $mahasiswa->nrp=$datanrp;
58      $mahasiswa->nama=$datanama;
59      $mahasiswa->fkmatkul=$datafk;
60      $mahasiswa->photo=$filename;
61      try {
62          $mahasiswa->save();
63      } catch (QueryException $e) {
64
65          return back()->withErrors(["status"=>"Fail to input data"]);
66      }
67      return redirect("/home");
68  }

```

Gambar 4.2. 30 kode program controller tambah mahasiswa

Gambar 4.2.30 adalah kode program yang telah penulis kerjakan pada *controller* untuk menambahkan data mahasiswa.

```

48     public function store(Request $request)
49     {
50         $dataidmatkul = $request->idmatkul;
51         $datamatkul = $request->matkul;
52         $datasks = $request->sks;
53         $datafksemester = $request->fksemester;
54         $matkuls = new matkul();
55         $matkuls->idmatkul=$dataidmatkul;
56         $matkuls->matkul=$datamatkul;
57         $matkuls->sks=$datasks;
58         $matkuls->fksemester=$datafksemester;
59         try {
60             $matkuls->save();
61         } catch (QueryException $e) {
62
63             return back()->withErrors(["status"=>"Fail to input data"]);
64         }
65         return redirect("/matkul");
66     }

```

Gambar 4.2. 31 kode program *controller* tambah mata kuliah

Gambar 4.2.31 adalah kode program yang telah penulis kerjakan pada *controller* untuk menambahkan data mata kuliah.

```

38     public function store(Request $request)
39     {
40         $dataidsemester = $request->idsemester;
41         $datasemester = $request->semester;
42         $semesters = new semester();
43         $semesters->idsemester=$dataidsemester;
44         $semesters->semester=$datasemester;
45
46         try {
47             $semesters->save();
48         } catch (QueryException $e) {
49
50             return back()->withErrors(["status"=>"Fail to input data"]);
51         }
52
53     }
54 }
```

Gambar 4.2. 32 kode program *controller* tambah semester

Gambar 4.2.32 adalah kode program yang telah penulis kerjakan pada *controller* untuk menambahkan data semester.

Mahasiswa

2072074	Siswa 1	Mata kuliah Keyshawn Russel
Choose file	Screenshot (2).png	
<input type="button" value="submit"/>		

Gambar 4.2. 33 tampilan tambah mahasiswa

Gambar 4.2.33 adalah tampilan untuk menambahkan data Mahasiswa. Pada gambar tersebut, penulis ingin menambahkan data mahasiswa baru. Selanjutnya klik “submit” untuk menambahkan data tersebut.

NRP	Nama	Mata Kuliah	Photo	action
2072016	Dr. Santos Huel PhD	Allene Turner		<button>Delete</button> <button>Edit</button>
2072043	Samara Lockman	Raphael Green		<button>Delete</button> <button>Edit</button>
2072074	Siswa 1	Keyshawn Russel		<button>Delete</button> <button>Edit</button>
2072094	Victor Hoppe	Raphael Green		<button>Delete</button> <button>Edit</button>
2072100	Percy Harber	Jordan Romaguera		<button>Delete</button> <button>Edit</button>

Gambar 4.2. 34 tampilan berhasil tambah mahasiswa

Gambar 4.2.34 adalah tampilan ketika data mahasiswa baru berhasil ditambahkan. Data mahasiswa baru tersebut langsung ditampilkan pada tabel Mahasiswa.

Mata Kuliah

Gambar 4.2. 35 tampilan tambah mata kuliah

Gambar 4.2.35 adalah tampilan untuk menambahkan data mata kuliah. Pada gambar tersebut, penulis ingin menambahkan data mata kuliah baru. Selanjutnya klik “Submit” untuk menambahkan data tersebut.

The screenshot shows a table titled "Mata Kuliah" with columns: ID Mata Kuliah, Mata Kuliah, Total SKS, Semester, and action (Delete, Edit). The table contains six rows of data. At the top, there are input fields for "ID Mata Kuliah" (IN1111), "Mata Kuliah" (Mata Kuliah 1), "Total SKS" (3), "Semester" (5), and a "Submit" button. Below the table, there is a search bar and a message indicating "Showing 1 to 6 of 6 entries". At the bottom right, there are "Previous", "1", and "Next" buttons.

ID Mata Kuliah	Mata Kuliah	Total SKS	Semester	action
IN1111	Mata Kuliah 1	3	5	<button>Delete</button> <button>Edit</button>
IN20101	Bradly Gorczany	1	9	<button>Delete</button> <button>Edit</button>
IN20102	Chester O'Conner	1	7	<button>Delete</button> <button>Edit</button>
IN20103	Guadalupe Lubowitz PhD	8	2	<button>Delete</button> <button>Edit</button>
IN20104	Veda Tromp	5	8	<button>Delete</button> <button>Edit</button>
IN20105	Agustina Cummerata Jr.	2	4	<button>Delete</button> <button>Edit</button>

Gambar 4.2. 36 tampilan mata kuliah berhasil tambah

Gambar 4.2.36 adalah tampilan ketika data mata kuliah baru berhasil ditambahkan. Data mata kuliah baru tersebut langsung ditampilkan pada tabel Mata Kuliah.

Semester

The screenshot shows a form for adding a new semester. It has two input fields: "1" and "1", and a "Submit" button.

Gambar 4.2. 37 tampilan tambah semester

Gambar 4.2.37 adalah tampilan untuk menambahkan data semester. Pada gambar tersebut, penulis ingin menambahkan data semester baru. Selanjutnya klik "Submit" untuk menambahkan data tersebut.

ID	Semester	action
1	1	Delete Edit
2	2	Delete Edit
3	3	Delete Edit
5	5	Delete Edit
6	6	Delete Edit
10	10	Delete Edit
11	11	Delete Edit

Gambar 4.2. 38 tampilan semester berhasil tambah

Gambar 4.2.38 adalah tampilan ketika data semester baru berhasil ditambahkan. Data semester baru tersebut langsung ditampilkan pada tabel Semester.

4.2.5 Update

Selanjutnya adalah membuat bagian untuk *update* atau mengubah data. Penulis terlebih dahulu membuat *route* untuk mengubah data untuk tabel mahasiswa, mata kuliah, dan semester.

```

27 |     Route::get('/home/edit/{mahasiswa}', [MahasiswaController::class, "edit"])
28 |     Route::get('/matkul/edit/{matkul}', [MatkulController::class, "edit"])
29 |     Route::get('/semester/edit/{semester}', [SemesterController::class, "edit"])
30 |     Route::patch('/home/edit/{mahasiswa}', [MahasiswaController::class, "update"]);
31 |     Route::patch('/matkul/edit/{matkul}', [MatkulController::class, "update"]);
32 |     Route::patch('/semester/edit/{semester}', [SemesterController::class, "update"]);

```

Gambar 4.2. 39 kode program *route update* projek sederhana

Gambar 4.2.39 adalah kode program yang telah penulis kerjakan untuk membuat *route update* untuk mengubah data mahasiswa, mata kuliah, dan semester. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *get* dan *post*, karena berfungsi untuk menampilkan dan mengubah data. Untuk “/home/edit/{mahasiswa}” akan mengubah data mahasiswa, untuk “/home/edit/{matkul}” akan mengubah data mata kuliah, untuk “/home/edit/{semester}” akan mengubah data semester. Untuk mengubah data,

maka fungsi yang akan dipanggil dalam *controller* bernama “update”. Untuk menampilkan data pada bagian select, maka fungsi yang akan dipanggil dalam *controller* bernama “edit”.

```

1  @extends('master')
2  @section("content")
3      <h1>Edit Mata Kuliah</h1>
4      {{Form::open(["method"=>"POST"])}}
5      {{Form::hidden("_method", "PATCH")}}
6      {{Form::text("idmatkul", $matkul->idmatkul, ["placeholder"=>"ID Matkul", "readonly"])}}
7      {{Form::text("matkul", $matkul->matkul, ["placeholder"=>"Mata kuliah"])}}
8      {{Form::select("skls", ["2"=>"2 SKS", "3"=>"3 SKS", "4"=>"4 SKS"], $matkul->skls)}}
9      {{Form::select("fksemester", $semester, $matkul->fksemester)}}
10     {{Form::submit("Update")}}
11     {{Form::close()}}
12 @endsection
13

```

Gambar 4.2. 40 kode program tampilan ubah mahasiswa

Gambar 4.2.40 adalah kode program yang kerjakan untuk membuat tampilan untuk mengubah data mahasiswa. Terdapat *input text* untuk mengubah data NRP dan Nama, kemudian terdapat *select* yang menggunakan fitur Select2, di mana *select* ini berisi mata kuliah, lalu terdapat *field* untuk meng-upload gambar, dan terdapat tombol “Update”.

```

1  @extends('master')
2  @section("content")
3      <h1>Edit Mata Kuliah</h1>
4      {{Form::open(["method"=>"POST"])}}
5      {{Form::hidden("_method", "PATCH")}}
6      {{Form::text("idmatkul", $matkul->idmatkul, ["placeholder"=>"ID Matkul", "readonly"])}}
7      {{Form::text("matkul", $matkul->matkul, ["placeholder"=>"Mata kuliah"])}}
8      {{Form::select("skls", ["2"=>"2 SKS", "3"=>"3 SKS", "4"=>"4 SKS"], $matkul->skls)}}
9      {{Form::select("fksemester", $semester)}}
10     {{Form::submit(["Update"])}}
11     {{Form::close()}}
12 @endsection
13

```

Gambar 4.2. 41 kode program tampilan ubah mata kuliah

Gambar 4.2.41 adalah kode program yang penulis kerjakan membuat tampilan untuk mengubah data mata kuliah. Terdapat *input text* untuk mengubah data ID Mata Kuliah dan Mata Kuliah, kemudian terdapat *select* yang menggunakan fitur Select2, di mana *select* ini berisi SKS, lalu terdapat *select* yang menggunakan fitur Select2, di mana *select* ini berisi Semester, dan terdapat tombol “Update”.

```

1  @extends('master')
2  @section("content")
3      <h1>Edit Semester</h1>
4      {{Form::open(["method"=>"POST"])}}
5      {{Form::hidden("_method","PATCH")}}
6      {{Form::text("idsemester",$semester->idsemester,[ "placeholder"=>"ID Semester"])}}
7      {{Form::text("semester",$semester->idsemester,[ "placeholder"=>"Semester"])}}
8      {{Form::submit("Update")}}
9      {{Form::close()}}
10     @endsection

```

Gambar 4.2. 42 kode program tampilan ubah semester

Gambar 4.2.42 adalah kode program yang penulis kerjakan membuat tampilan untuk mengubah data semester. Terdapat *input text* untuk mengubah data ID Semester dan Semester, kemudian terdapat tombol “Update”.

```

76  public function edit(mahasiswa $mahasiswa)
77  {
78      $matkuls = matkul::all();
79      foreach($matkuls as $item){
80          $matkulcoll[$item->idmatkul] = "Mata Kuliah ".$item->matkul;
81      }
82      return view("mahasiswa.edit",["mahasiswa"=>$mahasiswa,"matkul"=>$matkulcoll]);
83  }
84  public function update(Request $request, mahasiswa $mahasiswa)
85  {
86      $nama = $request->nama;
87      $matkul = $request->fkmatkul;
88      $mahasiswa->nama=$nama;
89      $mahasiswa->fkmatkul=$matkul;
90      $mahasiswa->update();
91      return redirect("/home");
92  }

```

Gambar 4.2. 43 kode program controller ubah mahasiswa

Gambar 4.2.43 adalah kode program yang telah penulis kerjakan pada *controller* untuk mengubah data mahasiswa.

```

79  public function edit(matkul $matkul)
80  {
81      $semesters = semester::all();
82      $semestercoll = [];
83      foreach($semesters as $item){
84          $semestercoll[$item->idsemester] = "Semester ".$item->semester;
85      }
86      return view("matkul.editmatkul",["matkul"=>$matkul,"semester"=>$semestercoll]);
87  }
88  public function update(Request $request, matkul $matkul)
89  {
90      $namamatkul = $request->matkul;
91      $nomatkul = $request->idmatkul;
92      $sksmatkul = $request->sks;
93      $semmatkul = $request->fksemester;
94      $matkul->matkul=$namamatkul;
95      $matkul->idmatkul=$nomatkul;
96      $matkul->sks=$sksmatkul;
97      $matkul->fksemester=$semmatkul;
98      $matkul->update();
99      return redirect("/matkul");
100 }
101 }
102 }
103 }

```

Gambar 4.2. 44 kode program controller mata kuliah

Gambar 4.2.44 adalah kode program yang telah penulis kerjakan pada *controller* untuk mengubah data mata kuliah.

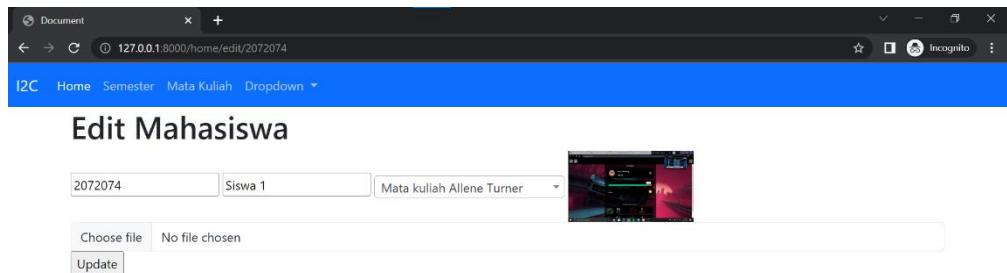
```

67  public function edit(semester $semester)
68  {
69      return view("semester.editsemester",["semester"=>$semester]);
70  }
71
72  public function update(Request $request, semester $semester)
73  [
74      $namasemester = $request->semester;
75      $nosemester = $request->idsemester;
76      $semester->semester=$namasemester;
77      $semester->idsemester=$nosemester;
78      $semester->update();
79      return redirect("/semester");
80  ]

```

Gambar 4.2. 45 kode program *controller* semester

Gambar 4.2.45 adalah kode program yang telah penulis kerjakan pada *controller* untuk mengubah data semester.



Gambar 4.2. 46 tampilan edit mahasiswa

Gambar 4.2.46 adalah tampilan ketika tombol “edit” diklik pada data mahasiswa yang baru saja ditambahkan. Pada gambar tersebut, penulis ingin mengubah mata kuliah dari mahasiswa tersebut. Selanjutnya klik tombol “Update” untuk mengubah data tersebut.

NRP	Nama	Mata Kuliah	Photo	action
2072016	Dr. Santos Huel PhD	Allene Turner		<button>Delete</button> <button>Edit</button>
2072043	Samara Lockman	Raphael Green		<button>Delete</button> <button>Edit</button>
2072074	Siswa 1	Allene Turner		<button>Delete</button> <button>Edit</button>
2072094	Victor Hoppe	Raphael Green		<button>Delete</button> <button>Edit</button>
2072100	Percy Harber	Jordan Romaguera		<button>Delete</button> <button>Edit</button>

Gambar 4.2.47 tampilan mahasiswa berhasil diubah

Gambar 4.2.47 adalah tampilan ketika data mahasiswa baru berhasil diubah. Terlihat bahwa mata kuliah dari mahasiswa tersebut telah berubah.

Gambar 4.2.48 tampilan edit mata kuliah

Gambar 4.2.48 adalah tampilan ketika tombol “edit” diklik pada data mata kuliah yang baru saja ditambahkan. Pada gambar tersebut, penulis ingin mengubah sks dan semester dari mata kuliah tersebut. Selanjutnya klik tombol “Update” untuk mengubah data tersebut.

ID Mata Kuliah	Mata Kuliah	Total SKS	Semester	action
IN11111	Mata Kuliah 1	4	6	<button>Delete</button> <button>Edit</button>
IN20101	Ms. Bernadette Oberbrunner DVM	5	4	<button>Delete</button> <button>Edit</button>
IN20102	Darren McKenzie	3	3	<button>Delete</button> <button>Edit</button>
IN20103	Dr. Dasia Kessler III	2	7	<button>Delete</button> <button>Edit</button>
IN20104	Brooke Bode	4	1	<button>Delete</button> <button>Edit</button>
IN20105	Miss Twila Langworth MD	6	10	<button>Delete</button> <button>Edit</button>

Gambar 4.2.49 tampilan mata kuliah berhasil diubah

Gambar 4.2.49 adalah tampilan ketika data mata kuliah baru berhasil berubah. Terlihat bahwa Total SKS dan Semester dari mata kuliah tersebut telah berubah.

Gambar 4.2.50 tampilan edit semester

Gambar 4.2.50 adalah tampilan ketika tombol “edit” diklik pada data semester yang baru saja ditambahkan. Pada gambar tersebut, penulis ingin mengubah id dan semester tersebut. Selanjutnya klik tombol “Update” untuk mengubah data tersebut.

ID	Semester	action
2	2	<button>Delete</button> <button>Edit</button>
5	5	<button>Delete</button> <button>Edit</button>
6	6	<button>Delete</button> <button>Edit</button>
7	7	<button>Delete</button> <button>Edit</button>
8	8	<button>Delete</button> <button>Edit</button>
9	9	<button>Delete</button> <button>Edit</button>

Gambar 4.2.51 tampilan semester berhasil diubah

Gambar 4.2.51 adalah tampilan ketika data semester baru berhasil berubah. Terlihat bahwa ID dan Semester tersebut telah berubah.

4.2.6 Delete

Selanjutnya adalah membuat bagian untuk *delete* atau menghapus data. Penulis terlebih dahulu membuat *route* untuk menghapus data untuk tabel mahasiswa, mata kuliah, dan semester.

```
42 |     Route::delete('/home/delete/{delnrp}', [MahasiswaController::class, "destroy"]);
43 |     Route::delete('/matkul/delete/{delidmatkul}', [MatkulController::class, "destroy"]);
44 |     Route::delete('/semester/delete/{delidsemester}', [SemesterController::class, "destroy"]);
```

Gambar 4.2. 52 kode program *route delete* projek sederhana

Gambar 4.2.52 adalah kode program yang telah penulis kerjakan untuk membuat *route delete* untuk menghapus data mahasiswa, mata kuliah, dan semester. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *delete*, karena berfungsi untuk menghapus data. Untuk “/home/delete/{delnrp}” akan menghapus data mahasiswa, untuk “/home/delete/{delidmatkul}” menghapus data mata kuliah, dan untuk “/home/delete/{delidsemester}” untuk menghapus data semester. Untuk menghapus data, maka fungsi yang akan dipanggil dalam *controller* bernama “destroy”.

```

55 <script>
56   $(document).ready(function() {
57     $('#table_id').DataTable();
58   });
59   function del(nrp) {
60     Swal.fire({
61       icon: "warning",
62       text: "Delete Data?",
63       showCancelButton: true
64     }).then((response) => {
65       if (response.isConfirmed) {
66         // window.location = '/home/delete/' + nrp
67         $.ajax({
68           url: '/home/delete/' + nrp,
69           type: 'POST',
70           data: {
71             '_method': 'DELETE',
72             '_token': '{{ csrf_token() }}'
73           },
74           success: (response) => {
75             if (response.status == "success") {
76               Swal.fire({
77                 icon: "success",
78                 text: "Deleted"
79               }).then((response)=>{
80                 location.reload();
81               })
82             }else{
83               Swal.fire({
84                 icon: "error",
85                 text: "Error"
86               }).then((response)=>{
87                 location.reload();
88               })
89             }
90           }
91         })
92       }
93     })
94   }
95 </script>

```

Gambar 4.2. 53 kode program tampilan hapus mahasiswa

Gambar 4.2.53 adalah kode program yang telah penulis kerjakan untuk tampilan menghapus data mahasiswa dengan menggunakan fitur dari SWAL (*Sweet Alert*). Ketika tombol “delete” diklik, maka akan muncul *alert* dengan pesan “Delete Data?” dan tombol “OK” serta tombol “Cancel”.

```

62   function del(idmatkul) {
63     // if(confirm('delete data?')){
64     //   window.location = '/home/delete/' + nrp
65     // }
66     Swal.fire({
67       icon: "warning",
68       text: "Delete Data?",
69       showCancelButton: true
70     }).then((response) => {
71       if (response.isConfirmed) {
72         // window.location = '/home/delete/' + nrp
73         $.ajax({
74           url: '/matkul/delete/' + idmatkul,
75           type: 'POST',
76           data: {
77             '_method': 'DELETE',
78             '_token': '{{ csrf_token() }}'
79           },
80           success: (response) => {
81             if (response.status == "success") {
82               Swal.fire({
83                 icon: "success",
84                 text: "Deleted"
85               }).then((response)=>{
86                 location.reload();
87               })
88             }else{
89               Swal.fire({
90                 icon: "error",
91                 text: "Error"
92               }).then((response)=>{
93                 location.reload();
94               })
95             }
96           }
97         })
98       }
99     })
100   }
101 
```

Gambar 4.2. 54 kode program tampilan hapus mata kuliah

Gambar 4.2.54 adalah kode program yang telah penulis kerjakan untuk tampilan menghapus data mata kuliah dengan menggunakan fitur dari SWAL (*Sweet Alert*). Ketika tombol “delete” diklik, maka akan muncul *alert* dengan pesan “Delete Data?” dan tombol “OK” serta tombol “Cancel”.

```

50 |     function del(idsemester) {
51 |         // if(confirm('delete data?')){
52 |             // window.location = '/home/delete/' + nrp
53 |             //
54 |         Swal.fire({
55 |             icon: "warning",
56 |             text: "Delete Data?",
57 |             showCancelButton: true
58 |         }).then((response) => {
59 |             if (response.isConfirmed) {
60 |                 // window.location = '/home/delete/' + nrp
61 |                 $.ajax({
62 |                     url: '/semester/delete/' + idsemester,
63 |                     type: 'POST',
64 |                     data: {
65 |                         '_method': 'DELETE',
66 |                         '_token': '{{ csrf_token() }}'
67 |                     },
68 |                     success: (response) => {
69 |                         if (response.status == "success") {
70 |                             Swal.fire({
71 |                                 icon: "success",
72 |                                 text: "Deleted"
73 |                             }).then((response)=>{
74 |                                 location.reload();
75 |                             })
76 |                         } else {
77 |                             Swal.fire({
78 |                                 icon: "error",
79 |                                 text: "Error"
80 |                             }).then((response)=>{
81 |                                 location.reload();
82 |                             })
83 |                         }
84 |                     }
85 |                 })
86 |             }
87 |         })
88 |     }

```

Gambar 4.2. 55 kode program tampilan hapus semester

Gambar 4.2.55 adalah kode program yang telah penulis kerjakan untuk tampilan menghapus data semester dengan menggunakan fitur dari SWAL (*Sweet Alert*). Ketika tombol “delete” diklik, maka akan muncul *alert* dengan pesan “Delete Data?” dan tombol “OK” serta tombol “Cancel”.

```

111 |     public function destroy(mahasiswa $delnrp)
112 |     {
113 |         $filename= public_path('/image/'). $delnrp->photo;
114 |         unlink($filename);
115 |         $result = $delnrp->delete();
116 |         // $delnrp->delete();
117 |         if($result){
118 |             return response()->json(["status"=>"success"]);
119 |         }else{
120 |             return response()->json(["status"=>"error"]);
121 |         }
122 |     }
123 |

```

Gambar 4.2. 56 kode program controller hapus mahasiswa

Gambar 4.2.56 adalah kode program yang telah penulis kerjakan pada *controller* untuk menghapus data mahasiswa.

```

108     public function destroy(matkul $delidmatkul)
109     {
110         $result = $delidmatkul->delete();
111         // $delidmatkul->delete();
112         if($result){
113             return response()->json(["status"=>"success"]);
114         }else{
115             return response()->json(["status"=>"error"]);
116         }
117     }
118 }
```

Gambar 4.2. 57 kode program controller hapus mata kuliah

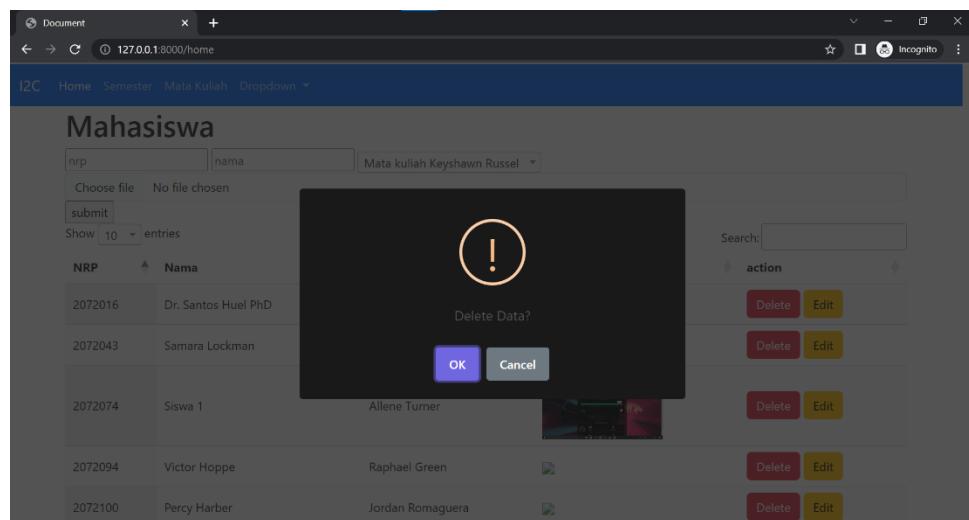
Gambar 4.2.6.57 adalah kode program yang telah penulis kerjakan pada *controller* untuk menghapus data mata kuliah.

```

85     public function destroy(semester $delidsemester)
86     {
87         $result = $delidsemester->delete();
88         // $delidsemester->delete();
89         if($result){
90             return response()->json(["status"=>"success"]);
91         }else{
92             return response()->json(["status"=>"error"]);
93         }
94     }
95 }
```

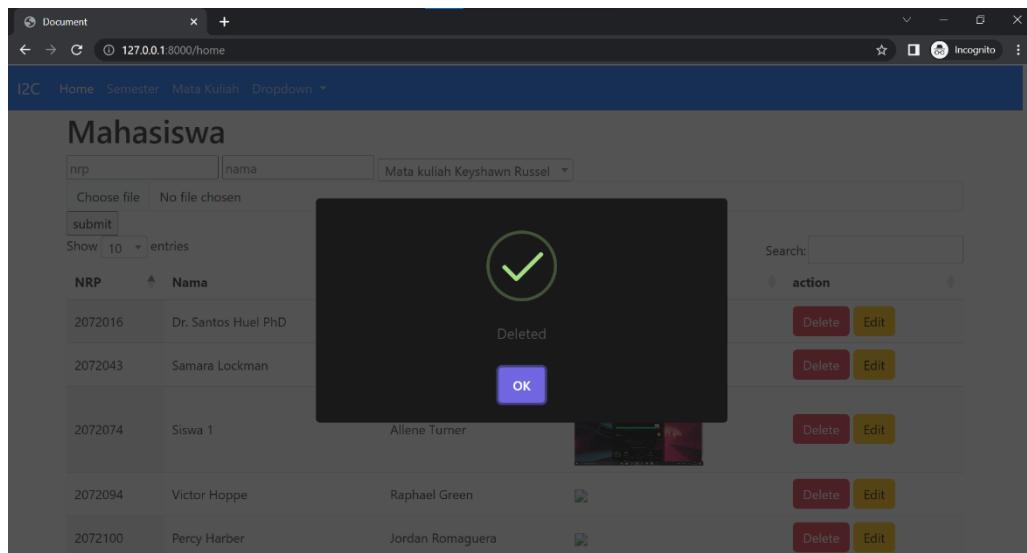
Gambar 4.2. 58 kode program controller hapus semester

Gambar 4.2.58 adalah kode program yang telah penulis kerjakan pada *controller* untuk menghapus data semester.



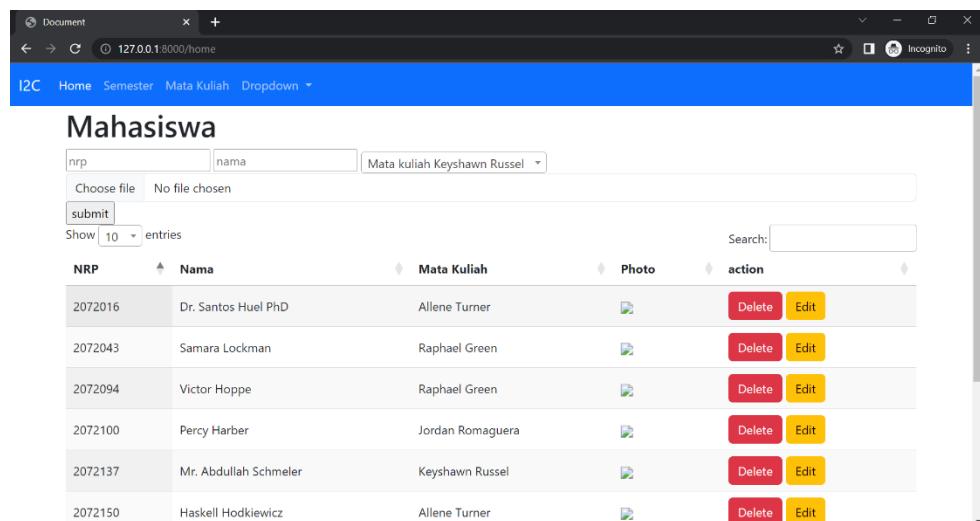
Gambar 4.2. 59 tampilan SWAL hapus mahasiswa

Gambar 4.2.59 adalah tampilan ketika tombol “delete” diklik pada mahasiswa baru. Ketika tombol “delete” diklik, akan muncul SWAL yang akan memunculkan konfirmasi sebelum data mahasiswa baru tersebut dihapus. Untuk menghapus data tersebut, selanjutnya klik tombol “OK” pada SWAL.



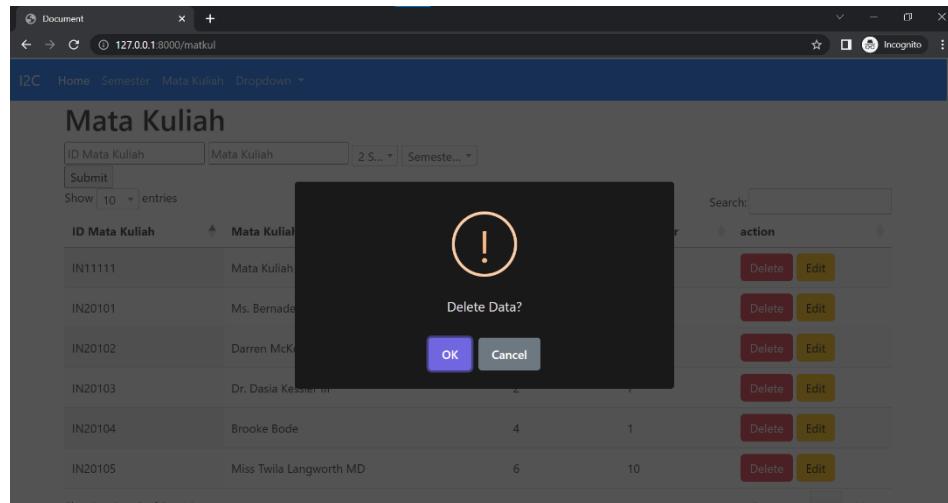
Gambar 4.2. 60 tampilan pesan mahasiswa dihapus

Gambar 4.2.60 adalah tampilan ketika tombol “OK” diklik. Akan muncul pesan bahwa data mahasiswa baru tersebut sudah terhapus. Selanjutnya klik “OK” untuk kembali ke halaman Mahasiswa.



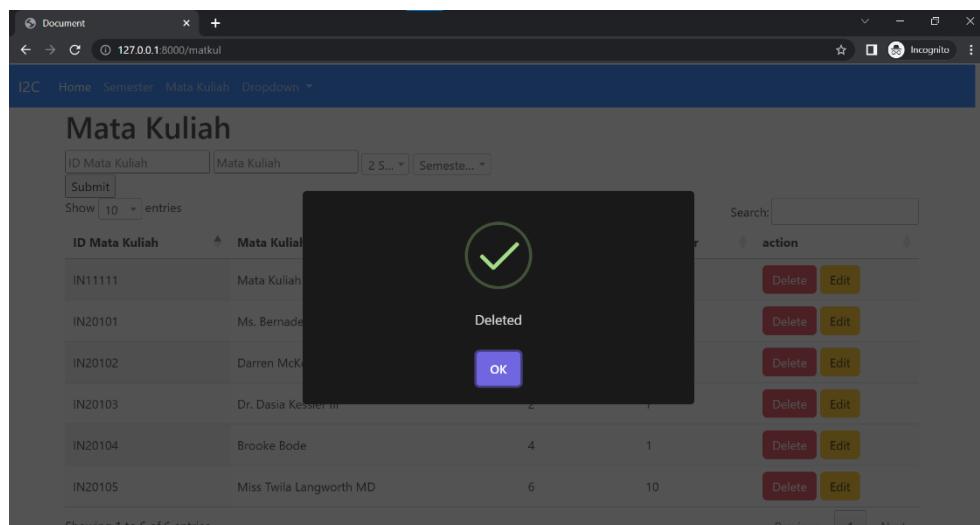
Gambar 4.2. 61 tampilan mahasiswa berhasil dihapus

Gambar 4.2.61 adalah tampilan ketika tombol “OK” sudah diklik. Setelah diklik, akan kembali ke halaman Mahasiswa. Data mahasiswa baru sudah terhapus dan tidak terdapat lagi pada tabel Mahasiswa.



Gambar 4.2. 62 tampilan Swal hapus mata kuliah

Gambar 4.2.62 adalah tampilan ketika tombol “delete” diklik pada mata kuliah baru. Ketika tombol “delete” diklik, akan muncul Swal yang akan memunculkan konfirmasi sebelum data mata kuliah baru tersebut dihapus. Untuk menghapus data tersebut, selanjutnya klik tombol “OK” pada Swal.



Gambar 4.2. 63 tampilan pesan mata kuliah dihapus

Gambar 4.2.63 adalah tampilan ketika tombol “OK” diklik. Akan muncul pesan bahwa data mata kuliah baru tersebut sudah terhapus. Selanjutnya klik “OK” untuk kembali ke halaman Mata Kuliah.

ID Mata Kuliah	Mata Kuliah	Total SKS	Semester	Action
IN20101	Ms. Bernadette Oberbrunner DVM	5	4	<button>Delete</button> <button>Edit</button>
IN20102	Darren McKenzie	3	3	<button>Delete</button> <button>Edit</button>
IN20103	Dr. Dasia Kessler III	2	7	<button>Delete</button> <button>Edit</button>
IN20104	Brooke Bode	4	1	<button>Delete</button> <button>Edit</button>
IN20105	Miss Twila Langworth MD	6	10	<button>Delete</button> <button>Edit</button>

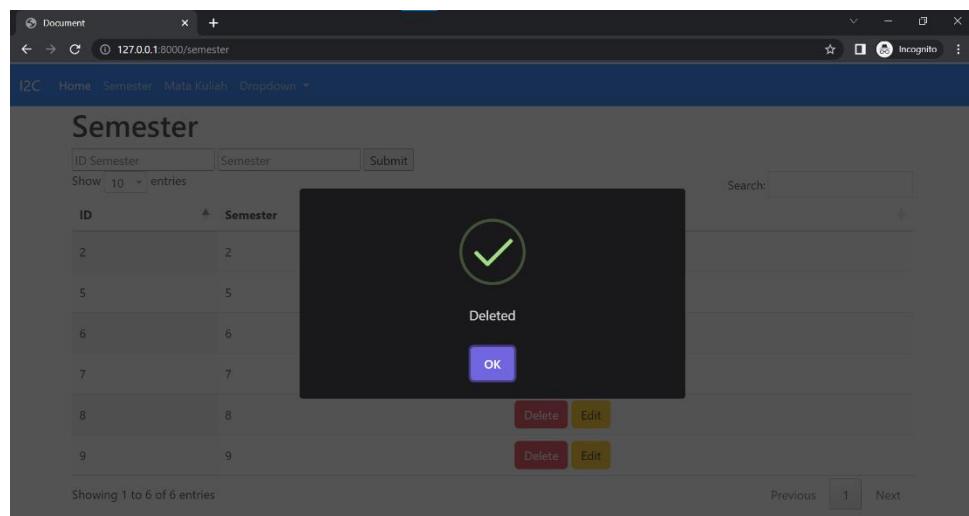
Gambar 4.2. 64 tampilan mata kuliah berhasil dihapus

Gambar 4.64 adalah tampilan ketika tombol “OK” sudah diklik. Setelah diklik, akan kembali ke halaman Mata Kuliah. Data mata kuliah baru sudah terhapus dan tidak terdapat lagi pada tabel Mata Kuliah.

ID Semester	Semester
2	2
5	5
6	6
7	7
8	8
9	9

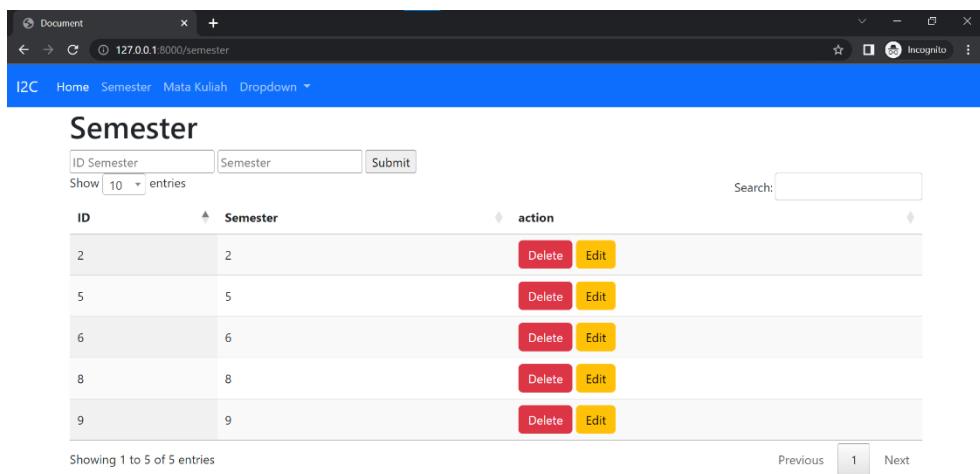
Gambar 4.2. 65 tampilan SWAL hapus Semester

Gambar 4.2.65 adalah tampilan ketika tombol “delete” diklik pada semester baru. Ketika tombol “delete” diklik, akan muncul SWAL yang akan memunculkan konfirmasi sebelum data semester baru tersebut dihapus. Untuk menghapus data tersebut, selanjutnya klik tombol “OK” pada SWAL.



Gambar 4.2. 66 tampilan pesan semester dihapus

Gambar 4.2.66 adalah tampilan ketika tombol “OK” diklik. Akan muncul pesan bahwa data semester baru tersebut sudah terhapus. Selanjutnya klik “OK” untuk kembali ke halaman Semester.



Gambar 4.2. 67 tampilan semester berhasil dihapus

Gambar 4.2.67 adalah tampilan ketika tombol “OK” sudah diklik. Setelah diklik, akan kembali ke halaman Semester. Data mata kuliah baru sudah terhapus dan tidak terdapat lagi pada tabel Semester.

4.3 Tes berupa *Mini Project*

Untuk menguji pengetahuan dan kemampuan penulis sebelum mengerjakan master data pada proyek SIPPMM, bapak Julio Narabel dari pihak I2C Studio memberikan tes kepada penulis berupa *mini project*. Pada mini project ini, sebetulnya tidak begitu jauh dengan apa yang telah penulis pelajari dan kerjakan pada bab sebelumnya. Pada *mini project* ini, terdapat 3 modul, yaitu modul Theater, modul Studio, dan modul Film. Ketiga modul tersebut memiliki tabel yang saling berelasi. Pertama-tama penulis akan menghubungkan PostgreSQL dengan *mini project* ini.

```

11 DB_CONNECTION=pgsql
12 DB_HOST=127.0.0.1
13 DB_PORT=5432
14 DB_DATABASE=tugas1
15 DB_USERNAME=postgres
16 DB_PASSWORD=

```

Gambar 4.3. 1 kode program .env *mini project*

Gambar 4.3.1 adalah kode program pada file .env untuk menghubungkan PostgreSQL dengan *mini project*. Nama database yang penulis buat pada PostgreSQL untuk mini project adalah “tugas1”. Username yang digunakan adalah “postgres” sesuai dengan username pada PostgreSQL.

4.3.1 Master Layout

Selanjutnya adalah membuat *master* untuk *layout*. Ini dilakukan untuk membuat master dari tampilan web. Dalam master ini, akan terdapat kode program untuk *import/install* fitur-fitur yang nantinya akan digunakan.

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8     <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/v/ct/jq-3.6.0/dt-1.13.2/datatables.min.css" />
9     <script type="text/javascript" src="https://cdn.datatables.net/v/ct/jq-3.6.0/dt-1.13.2/datatables.min.js"></script>
10    <!-- Bootstrap -->
11    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-shJNbQB1Q8Bd1163oM4atOpb7j1n13/r596d6611kw/cmA6l8g0" crossorigin="anonymous">
12    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-y6QfDfWFOzvXwYQF1y0H1ZmXu2Ig1wXVqF7CWh" crossorigin="anonymous"></script>
13    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
14    <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
15    <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js" integrity="sha512-kqIFES+RuoG0B3e982ELVRYwUQvNvPTItv3D6ujphhtGKvd7BqdKkqvP2T93jg0" rel="stylesheet" href="https://code.claudiore.com/sgx/lib/select2-bootstrap-theme/4.1.0-beta.10/select2-bootstrap-theme.min.css" />
16  </head>
17  <body>
18    <nav class="navbar navbar-expand-lg fixed-top bg-dark" data-bs-theme="dark">
19      <div class="container-fluid">
20        <a class="navbar-brand" href="/theater">T2C</a>
21        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
22          <span class="navbar-toggler-icon"></span>
23        </button>
24        <div class="collapse navbar-collapse" id="navbarSupportedContent">
25          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
26            <li class="nav-item">
27              <a class="nav-link active" aria-current="page" href="/theater">Bioskop</a>
28            </li>
29            <li class="nav-item">
30              <a class="nav-link" href="#">Film</a>
31            </li>
32            <li class="nav-item">
33              <a class="nav-link" href="#">Studio</a>
34            </li>
35          </ul>
36        </div>
37      </div>
38    </nav>
39    <div class="container">
40      <div id="content">
41        </div>
42        <script>
43          $(document).ready(function() {
44            $('#table_id').DataTable();
45          });
46        </script>
47      </div>
48    </body>
49  </html>

```

Gambar 4.3.2 kode program *master layout mini project*

Gambar 4.3.2 adalah kode program yang penulis kerjakan untuk membuat *master layout* pada *mini project*. Pada kode program tersebut terdapat kode program untuk melakukan *import/install* CDN terhadap fitur-fitur atau *library* yang akan digunakan, seperti Bootstraps, DataTables, SWAL, dan Select2. Terdapat juga kode program untuk membuat *navigation bar*. Untuk dapat menggunakan master layout ini, maka pada tampilan atau view perlu menambahkan kode program “extends(“master”)” dan “section(“content”)”.

4.3.2 Migrasi dan Model

Selanjutnya adalah membuat migrasi dan model untuk tabel *theaters*, tabel *studios*, dan tabel *films*. Terdapat beberapa *task* untuk membuat migrasi dan model pada *mini project* ini.

4.3.2.1 Tabel theaters

Tabel theaters

- Nama Migrasi: create_theaters_table.
- Nama Tabel: theaters.
- Rincian Tabel:
 - id
 - nama_bioskop => string => unique
 - kota => string
 - jumlah_studio => integer => default 0
 - timestamps()
 - softDeletes()
- Model:
 - Nama Model: Theater.
 - Sediakan const KOTA = ['Bandung', 'Jakarta', 'Semarang', 'Surabaya', 'Yogyakarta'].

Gambar 4.3. 3 task migrasi dan model tabel theaters

Gambar 4.3.3 adalah *task* untuk membuat migrasi dan model untuk tabel theaters. Pada *task* ini, penulis membuat migrasi dan model untuk tabel theaters.

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10      * Run the migrations.
11      */
12      public function up(): void
13      {
14          Schema::create('theaters', function (Blueprint $table) {
15              $table->id();
16              $table->string("nama_bioskop")->unique();
17              $table->string("kota");
18              $table->integer("jumlah_studio")->default(0);
19              $table->timestamps();
20              $table->softDeletes();
21          });
22      }
23
24      /**
25      * Reverse the migrations.
26      */
27      public function down(): void
28      {
29          Schema::dropIfExists('theaters');
30      }
31  };

```

Gambar 4.3. 4 kode program migrasi tabel theaters

Gambar 4.3.4 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel theaters. Terdapat kolom id, nama_bioskop, kota, dan jumlah_studio. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut dibuat. SoftDeletes adalah fitur dari Laravel untuk menghapus data sementara.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use App\Models\Studio;
8  use Illuminate\Database\Eloquent\SoftDeletes;
9
10 class Theater extends Model
11 {
12     use HasFactory;
13     use SoftDeletes;
14     protected $fillable = [
15         'nama_bioskop',
16         'kota',
17         'jumlah_studio',
18     ];
19     protected $table = "theaters";
20     const KOTA = ['Bandung', 'Jakarta', 'Semarang', 'Surabaya', 'Yogyakarta'];
21     public function theater_studio(){
22         return $this->hasMany(Studio::class,"theater_id");
23     }
24 }

```

Gambar 4.3. 5 kode program model tabel theaters

Gambar 4.3.5 adalah kode program untuk membuat model untuk tabel theaters. *Protected fillable* adalah kolom-kolom yang dapat diisi, yaitu kolom nama_bioskop, kota, dan jumlah_studio. Pada model ini juga terdapat *const KOTA* yang berisi daftar kota yang nantinya akan dimunculkan pada *select* pada saat akan menambah atau mengubah data *theater*. Terdapat juga relasi yang diperlukan untuk tabel theaters.

4.3.2.2 Tabel studios

Tabel studios

- Nama Migrasi: create_studios_table.
- Nama Tabel: studios.
- Rincian Tabel:
 - id
 - theater_id => unsigned integer => Foreign Key ke tabel theaters.
 - jenis_studio => string
 - studio => integer
 - timestamps()
 - softDeletes()
 - Gabungan / kombinasi antara kolom theater_id dan studio diset unique (composite index).

Gambar 4.3. 6 task migrasi tabel studios

Gambar 4.3.6 adalah *task* untuk membuat migrasi untuk tabel studios. Pada *task* ini, penulis membuat migrasi untuk tabel studios.

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10     * Run the migrations.
11     */
12    public function up(): void
13    {
14        Schema::create('studios', function (Blueprint $table) {
15            $table->id();
16            $table->unsignedInteger("theater_id")->foreign()->references("id")->on("theaters");
17            $table->foreign("theater_id")->references("id")->on("theaters");
18            $table->string("jenis_studio");
19            $table->integer("studio");
20            $table->timestamps();
21            $table->softDeletes();
22            $table->unique(["theater_id", "studio"]);
23        });
24    }
25
26    /**
27     * Reverse the migrations.
28     */
29    public function down(): void
30    {
31        Schema::dropIfExists('studios');
32    }
33 };

```

Gambar 4.3. 7 kode program migrasi tabel studios

Gambar 4.3.7 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel studios. Terdapat kolom id, theater_id, jenis_studio, dan studio. Untuk kolom theater_id dan studio dibuat menjadi kombinasi unik. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut dibuat. SoftDeletes adalah fitur dari Laravel untuk menghapus data sementara.

- Model:
 - Nama Model: Studio.
 - Sediakan const JENIS = ['basic', '3D', '4DX', 'starium', 'IMAX'].
- Relasi di Model Studio:
 - belongsTo:
 - Relasi: studio_theater.
 - Kolom Terkait: theater_id.
 - Model Target: Theater.
- Relasi di Model Theater:
 - hasMany:
 - Relasi: theater_studio.
 - Kolom Terkait: theater_id.
 - Model Target: Studio.



- Relasi: theater_studio.
- Kolom Terkait: theater_id.
- Model Target: Studio.

Gambar 4.3. 8 task model tabel studios

Gambar 4.3.8 adalah *task* untuk membuat model untuk tabel studios. Pada *task* ini, penulis membuat model untuk tabel studios.

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Theater;
8 use App\Models\Film;
9 use Illuminate\Database\Eloquent\SoftDeletes;
10
11 class Studio extends Model
12 {
13     use HasFactory;
14     use SoftDeletes;
15     protected $fillable = [
16         'jenis_studio',
17         'studio',
18         'theater_id'
19     ];
20
21     protected $table = "studios";
22     const JENIS = ['basic', '3D', '4DX', 'starium', 'IMAX'];
23     public function studio_theater(){
24         return $this->belongsTo(Theater::class,"theater_id");
25     }
26     public function studio_film(){
27         return $this->belongsToMany(Film::class,"studios_films","studio_id","film_id");
28     }
29 }
```

Gambar 4.3. 9 kode program model tabel studios

Gambar 4.3.9 adalah kode program untuk membuat model untuk tabel studios. *Protected fillable* adalah kolom-kolom yang dapat diisi, yaitu kolom jenis_studio, studio, dan theater_id. Pada model ini juga terdapat const JENIS yang berisi daftar jenis studio yang nantinya akan dimunculkan pada *select* pada saat akan menambah atau mengubah data *studio*. Terdapat juga relasi yang diperlukan untuk tabel studios.

4.3.2.3 Tabel films

Tabel films

- Nama Migrasi: create_films_table.
- Nama Tabel: films.
- Rincian Tabel:
 - id
 - film => string => unique
 - durasi => integer
 - rating => string
 - sinopsis => text => nullable
 - timestamps()
 - softDeletes()
- Model:
 - Nama Model: Film.
 - Sediakan const RATING = ['SU', 'PG-13', 'R'].

Gambar 4.3. 10 *task* migrasi dan model tabel films

Gambar 4.3.10 adalah *task* untuk membuat migrasi dan model untuk tabel *films*. Pada *task* ini, penulis membuat migrasi dan model untuk tabel *films*.

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('films', function (Blueprint $table) {
16             $table->id();
17             $table->string("film")->unique();
18             $table->integer("durasi");
19             $table->string("rating");
20             $table->text("sinopsis")->nullable();
21             $table->timestamps();
22             $table->softDeletes();
23         });
24
25     /**
26      * Reverse the migrations.
27      */
28     public function down(): void
29     {
30         Schema::dropIfExists('films');
31     }
32 };
33

```

Gambar 4.3. 11 kode program migrasi tabel films

Gambar 4.3.11 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel *films*. Terdapat kolom id, film, durasi, rating, dan sinopsis. Untuk kolom theater_id dan studio dibuat menjadi kombinasi unik. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut dibuat. SoftDeletes adalah fitur dari Laravel untuk menghapus data sementara.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use App\Models\FilmDetail;
8  use App\Models\Studio;
9  use Illuminate\Database\Eloquent\SoftDeletes;
10
11 class Film extends Model
12 {
13     use HasFactory;
14     use SoftDeletes;
15     protected $fillable = [
16         'film',
17         'durasi',
18         'rating',
19         'sinopsis'
20     ];
21     protected $table = "films";
22     const RATING = ['SU', 'PG-13', 'R'];
23     public function film_filmdetail(){
24         return $this->hasOne(FilmDetail::class,"film_id");
25     }
26     public function film_studio(){
27         return $this->belongsToMany(Studio::class,"studios_films","film_id","studio_id");
28     }
29 }

```

Gambar 4.3. 12 kode program model tabel films

Gambar 4.3.12 adalah kode program untuk membuat model untuk tabel films. *Protected fillable* adalah kolom-kolom yang dapat diisi, yaitu kolom film, durasi, rating, dan sinopsis. Pada model ini juga terdapat *const RATING* yang berisi daftar rating yang nantinya akan dimunculkan pada *select* pada saat akan menambah atau mengubah data *film*. Terdapat juga relasi yang diperlukan untuk tabel films.

4.3.2.4 Tabel films_details

Tabel films_details

- Nama Migrasi: `create_films_details_table`.
- Nama Tabel: `films_details`.
- Rincian Tabel:
 - `film_id => unsigned integer => Primary Key => Foreign Key ke tabel films.`
 - `jadwal_rilis => date.`
 - `bahasa => string.`
 - `timestamps()`
 - `softDeletes()`
- Model:
 - Nama Model: `FilmDetail`.
 - Sediakan const `BAHASA = ['indonesia', 'inggris', 'korea', 'thailand']`.
- Relasi di Model `FilmDetail`:
 - `belongsTo:`
 - Relasi: `filmdetail_film`.
 - Kolom Terkait: `film_id`.
 - Model Target: `Film`.
- Relasi di Model `Film`:
 - `hasOne:`
 - Relasi: `film_filmdetail`.
 - Kolom Terkait: `film_id`.
 - Model Target: `FilmDetail`.

Gambar 4.3. 13 task migrasi dan model tabel films_details

Gambar 4.3.13 adalah *task* untuk membuat migrasi dan model untuk tabel `films_details`. Pada *task* ini, penulis membuat migrasi dan model untuk tabel `films_details`.

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10      * Run the migrations.
11      */
12      public function up(): void
13      {
14          Schema::create('films_details', function (Blueprint $table) {
15              $table->unsignedInteger('film_id')->primary();
16              $table->foreign("film_id")->references("id")->on("films");
17              $table->date("jadwal_rilis");
18              $table->string("bahasa");
19              $table->timestamps();
20              $table->softDeletes();
21          });
22      }
23
24      /**
25      * Reverse the migrations.
26      */
27      public function down(): void
28      {
29          Schema::dropIfExists('films_details');
30      }
31  };
32

```

Gambar 4.3. 14 kode program migrasi tabel films_details

Gambar 4.3.14 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel films_details. Terdapat kolom film_id, jadwal_rilis, dan bahasa. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut dibuat. SoftDeletes adalah fitur dari Laravel untuk menghapus data sementara.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use App\Models\Film;
8  use Illuminate\Database\Eloquent\SoftDeletes;
9
10 class FilmDetail extends Model
11 {
12     use HasFactory;
13     use SoftDeletes;
14     protected $fillable = [
15         'jadwal_rilis',
16         'bahasa',
17         'film_id'
18     ];
19     protected $primaryKey = "film_id";
20     const BAHASA = ['indonesia', 'inggris', 'korea', 'thailand'];
21     protected $table = "films_details";
22     public function filmdetail_film(): [
23         return $this->belongsTo(Film::class, "film_id");
24     ]
25 }

```

Gambar 4.3. 15 kode program model tabel films_details

Gambar 4.3.15 adalah kode program untuk membuat model untuk tabel films_details. *Protected fillable* adalah kolom-kolom yang dapat diisi, yaitu kolom

jadwal_rilis, bahasa, dan film_id. Pada model ini juga terdapat *const BAHASA* yang berisi daftar bahasa yang nantinya akan dimunculkan pada *select* pada saat akan menambah atau mengubah data *detail film*. Terdapat juga relasi yang diperlukan untuk tabel films.

4.3.2.5 Tabel studios_films

Tabel studios_films

- Nama Migrasi: `create_studios_films_table`.
- Nama Tabel: `studios_films`.
- Rincian Tabel:
 - `studio_id => unsigned integer => Foreign Key ke tabel studios`.
 - `film_id => unsigned integer => Foreign Key ke tabel films`.
 - `timestamps()`
 - Set kolom `studio_id` dan `film_id` agar menjadi composite Primary Key, utk kebutuhan relasi Many to Many.
- Relasi di Model Studio:
 - `belongsToMany:`
 - Relasi: `studio_film`.
 - Kolumn Terkait: `film_id`.
 - Model Target: `Film`.
- Relasi di Model Film:
 - `belongsToMany:`
 - Relasi: `film_studio`.
 - Kolumn Terkait: `studio_id`.
 - Model Target: `Studio`.

Gambar 4.3. 16 task migrasi tabel studios_films

Gambar 4.3.16 adalah *task* untuk membuat migrasi untuk tabel `studios_films`. Pada *task* ini, penulis membuat migrasi untuk tabel `studios_films`. Tabel ini sebenarnya adalah tabel yang terbentuk karena tabel `studios` memiliki relasi *many to many* dengan tabel `films`.

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10      * Run the migrations.
11      */
12      public function up(): void
13      {
14          Schema::create('studios_films', function (Blueprint $table) {
15              $table->unsignedInteger("studio_id")->index();
16              $table->foreign("studio_id")->references("id")->on("studios");
17              $table->unsignedInteger("film_id")->index();
18              $table->foreign("film_id")->references("id")->on("films");
19              $table->timestamps();
20              $table->primary(["studio_id", "film_id"]);
21          });
22      }
23
24      /**
25      * Reverse the migrations.
26      */
27      public function down(): void
28      {
29          Schema::dropIfExists('studios_films');
30      }
31  };
32

```

Gambar 4.3. 17 kode program migrasi tabel studios_films

Gambar 4.3.17 adalah kode program yang telah penulis kerjakan untuk membuat migrasi untuk tabel studios_films. Terdapat kolom studio_id, dan film_id. Untuk kolom studio_id dan film_id dibuat menjadi kombinasi *primary*. Timestamps adalah kolom yang dibuat oleh Laravel di mana kolom ini akan mencatat waktu data tersebut dibuat. SoftDeletes adalah fitur dari Laravel untuk menghapus data sementara.

```

26  public function studio_film(){
27      return $this->belongsToMany(Film::class,"studios_films","studio_id","film_id");
28  }

```

Gambar 4.3. 18 kode program relasi di model studio

Gambar 4.3.18 adalah kode program yang telah penulis kerjakan untuk membuat relasi antara tabel studios dengan tabel films di model Studio.

```

26  public function film_studio(){
27      return $this->belongsToMany(Studio::class,"studios_films","film_id","studio_id");
28  }

```

Gambar 4.3. 19 kode program relasi di model film

Gambar 4.3.19 adalah kode program yang telah penulis kerjakan untuk membuat relasi antara tabel films dengan tabel studios di model Film.

Selanjutnya adalah membuat CRUD (*Create, Read Update, Delete*) untuk modul Theater, Studio, dan Film.

4.3.3 CRUD Theater

Untuk modul Theater pertama-tama penulis membuat untuk bagian *read* dahulu, kemudian *create*, *update*, dan terakhir *delete*.

4.3.3.1 Read

Modul Theater

Main Page

- Route: /theater
- Controller: TheaterController@index_theater
- Method: GET
- View: views/theater.blade.php.
- Tampilan:
 - Format tampilan dibuat kurang lebih seperti ini:

- Tulisan Judul: "Bioskop".
- Tombol di sebelah kanan atas: Tombol "Tambah". Ketika di-klik, munculkan Modal "Tambah Bioskop". Lihat bagian "Insert Theater".
- Rincian Tabel:
 - GET data dari tabel theaters, order by nama_bioskop ASCENDING.
 - # Bioskop Kota Jumlah Studio Aksi
 - Kolom "#" => diisi oleh urutan mulai dari 1 s/d jumlah data.
 - Kolom "Bioskop" => theaters->nama_bioskop.
 - Kolom "Kota" => theaters->kota.
 - Kolom "Jumlah Studio" => theaters->jumlah_studio.
 - Kolom "Aksi":
 - Tombol "Edit".
 - Sertakan atribut data-th = {theaters dari data ybs}.
 - Ketika tombol di-klik, munculkan Modal "Edit Bioskop". Lihat bagian "Update Theater".
 - Tombol "Hapus".
 - Sertakan atribut data-id = {theaters->id dari data ybs}.
 - Lihat bagian "Delete Theater".

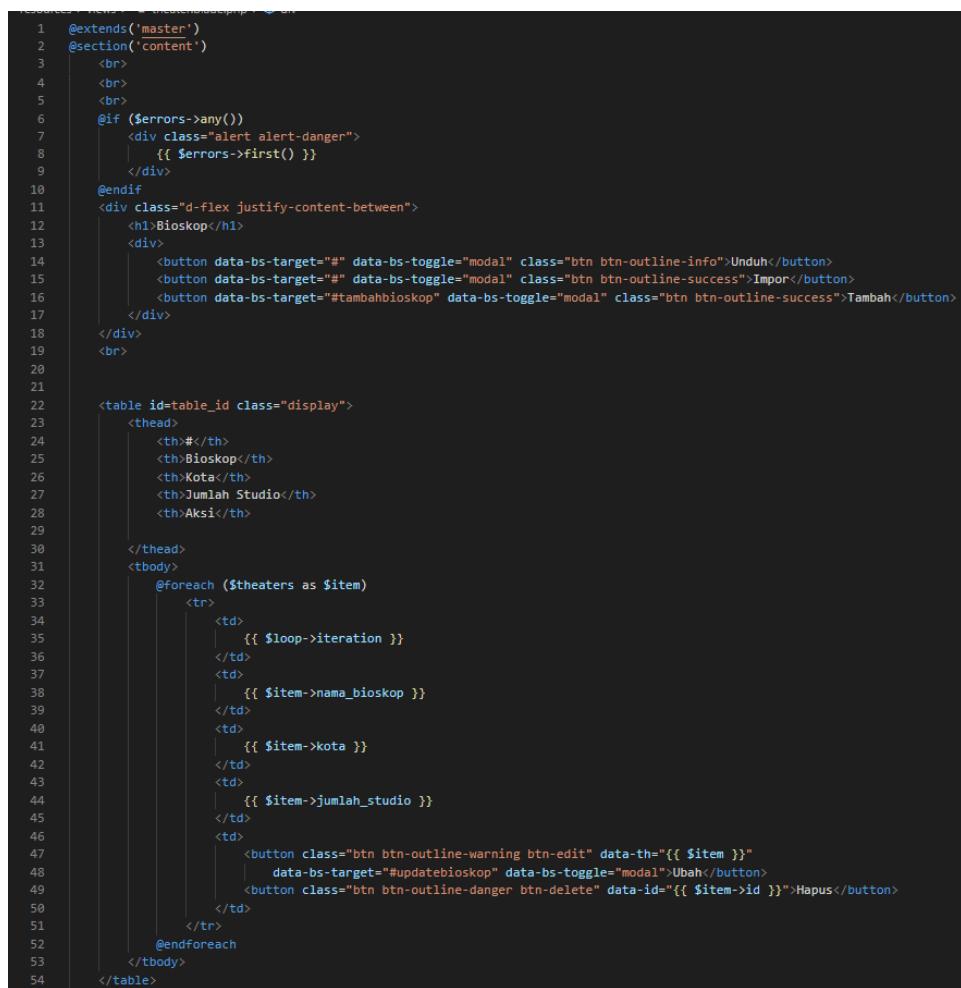
Gambar 4.3. 20 task main page / read Theater

Gambar 4.3.20 adalah *task* untuk membuat *main page* untuk modul Theater.

```
20   Route::GET('/theater', [TheaterController::class, "index_theater"]);
```

Gambar 4.3. 21 kode program route read Theater

Gambar 4.3.21 adalah kode program yang telah penulis kerjakan untuk membuat *route read* untuk menampilkan data modul Theater. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *get*, karena berfungsi untuk menampilkan data. Untuk menampilkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “index_theater”.



```
1 @extends('master')
2 @section('content')
3 <br>
4 <br>
5 <br>
6 @if ($errors->any())
7   <div class="alert alert-danger">
8     {{ $errors->first() }}
9   </div>
10 @endif
11 <div class="d-flex justify-content-between">
12   <h1>Bioskop</h1>
13   <div>
14     <button data-bs-target="#" data-bs-toggle="modal" class="btn btn-outline-info">Unduh</button>
15     <button data-bs-target="#" data-bs-toggle="modal" class="btn btn-outline-success">Import</button>
16     <button data-bs-target="#tambahbioskop" data-bs-toggle="modal" class="btn btn-outline-success">Tambah</button>
17   </div>
18 </div>
19 <br>
20
21
22 <table id=table_id class="display">
23   <thead>
24     <th>*</th>
25     <th>Bioskop</th>
26     <th>Kota</th>
27     <th>Jumlah Studio</th>
28     <th>Aksi</th>
29
30   </thead>
31   <tbody>
32     @foreach ($theaters as $item)
33       <tr>
34         <td>
35           {{ $loop->iteration }}
36         </td>
37         <td>
38           {{ $item->nama_bioskop }}
39         </td>
40         <td>
41           {{ $item->kota }}
42         </td>
43         <td>
44           {{ $item->jumlah_studio }}
45         </td>
46         <td>
47           <button class="btn btn-outline-warning btn-edit" data-th="{{ $item }}"
48             data-bs-target="#updatebioskop" data-bs-toggle="modal">Ubah</button>
49           <button class="btn btn-outline-danger btn-delete" data-id="{{ $item->id }}>Hapus</button>
50         </td>
51       </tr>
52     @endforeach
53   </tbody>
54 </table>
```

Gambar 4.3. 22 kode program tampilan Theater

Gambar 4.3.22 adalah kode program yang penulis kerjakan untuk membuat tampilan dari modul Theater. Pada tampilan ini akan menampilkan tabel yang menggunakan DataTables. Tabel ini akan menampilkan data-data dari tabel

theaters. Terdapat juga tombol Ubah untuk mengubah data theater tersebut dan Hapus untuk menghapus data theater tersebut.

```

12  public function index_theater()
13  {
14      $theaters = Theater::orderBy("nama_bioskop","ASC")->get();
15      $kotacoll = [];
16      foreach(Theater::KOTA as $item){
17          $kotacoll[$item]=$item;
18      }
19      return view("theater",["theaters"=>$theaters,"kotacoll"=>$kotacoll]);
20  }

```

Gambar 4.3. 23 kode program controller read Theaters

Gambar 4.3.23 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data dari tabel theaters. Data tersebut diurutkan berdasarkan abjad dari kolom nama_bioskop secara *ascending* atau menurun.

Tabel theaters saat ini masih kosong atau belum ada data, sehingga penulis perlu mengisi data *dummy* ke dalam tabel tersebut. Untuk mengisi data *dummy* ke dalam tabel theaters, penulis menggunakan salah satu fitur dari Laravel yaitu *seeder*

```

1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class TheaterSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         //
17         DB::table('theaters')->insert([
18             "nama_bioskop"=>"CGV TSM",
19             "kota"=>"Bandung",
20             "jumlah_studio"=>0,
21         ]);
22         DB::table('theaters')->insert([
23             "nama_bioskop"=>"XXI TSM",
24             "kota"=>"Bandung",
25             "jumlah_studio"=>0,
26         ]);
27         DB::table('theaters')->insert([
28             "nama_bioskop"=>"Asik",
29             "kota"=>"Bandung",
30             "jumlah_studio"=>0,
31         ]);
32
33     }
34 }

```

Gambar 4.3. 24 kode program seeder Theater

Gambar 4.3.24 adalah kode program yang telah penulis kerjakan pada bagian *seeder*. Kolom pada tabel theaters yang perlu diisi data *dummy* adalah kolom nama_bioskop, kolom kota, dan kolom jumlah_studio.

#	Bioskop	Kota	Jumlah Studio	Aksi
1	Asik	Bandung	0	<button>Ubah</button> <button>Hapus</button>
2	CGV TSM	Bandung	0	<button>Ubah</button> <button>Hapus</button>
3	XXI TSM	Bandung	0	<button>Ubah</button> <button>Hapus</button>

Gambar 4.3. 25 tampilan modul Theater

Gambar 4.3.25 adalah tampilan dari modul Theater. Tampilan ini memiliki judul “Bioskop”. Lalu terdapat tabel yang berisi kolom untuk penomoran, Bioskop, Kota, Jumlah Studio, serta Aksi di mana kolom ini berisi tombol Ubah dan Hapus. Selain itu terdapat juga tombol Tambah menampilkan modal untuk untuk menambah data Theater.

4.3.3.2 Create

Insert Theater

- Munculkan Modal "Tambah Bioskop" yg berisikan form dgn fields sbb:
 - Catatan: Terapkan LaravelCollective pada penggunaan Form.**
 - Input text "Nama Bioskop". Required.
 - Select "Kota".
 - Data / item diisi oleh const KOTA dari Model Theater.
 - Value = Teks = item dari const KOTA.
 - Placeholder: "Pilih Kota".



- Required.
- Pada Modal Footer:
 - Tombol "Batal". Ketika di-klik, CLOSE Modal.
 - Tombol "Simpan". Ketika di-klik, INSERT data ke database.
- INSERT data ke database:
 - Route: /theater
 - Controller: TheaterController@store_theater
 - Method: POST
 - Rincian di Controller:
 - Step 1**
 - Insert data ke tabel theaters:
 - nama_bioskop = value input "Nama Bioskop".
 - kota = value select "Kota" yg dipilih.
 - jumlah_studio = 0.
 - Step 2**
 - Return redirect ke halaman /theater.

Gambar 4.3. 26 task create Theater

Gambar 4.3.26 adalah *task* untuk membuat bagian *create* atau *insert* pada modul Theater.

```
21   Route::POST('/theater', [TheaterController::class, "store_theater"]);
```

Gambar 4.3. 27 kode program *route create* Theater

Gambar 4.3.27 adalah kode program yang telah penulis kerjakan untuk membuat *route create* untuk menambahkan data modul Theater. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *post*, karena berfungsi untuk menambahkan data. Untuk menambahkan data, maka fungsi yang akan dipanggil dalam *controller* bernama "store_theater".

```

55     <div class="modal fade" tabindex="-1" id="tambahbioskop">
56         <div class="modal-dialog">
57             <div class="modal-content">
58                 <div class="modal-header">
59                     <h5 class="modal-title">Tambah Bioskop</h5>
60                     <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
61                 </div>
62                 <div class="modal-body">
63                     {{ Form::open(['method' => 'POST', 'url' => '/theater']) }}
64                     <div class="mb-3">
65                         {{ Form::text('nama bioskop', null, ['class' => 'form-control', 'placeholder' => 'Nama Bioskop', 'required']) }}
66                     </div>
67                     <div class="mb-3">
68                         {{ Form::select('kota', $kotacoll, null, ['class' => 'form-select', 'placeholder' => 'Kota', 'required']) }}
69                     </div>
70                     <div class="modal-footer">
71                         {{ Form::reset('Batal', ['class' => 'btn btn-outline-secondary', 'data-bs-dismiss' => 'modal']) }}
72                         {{ Form::submit('Simpan', ['class' => 'btn btn-outline-primary']) }}
73                     </div>
74                     {{ Form::close() }}
75                 </div>
76             </div>
77         </div>
78     </div>

```

Gambar 4.3. 28 kode program tampilan tambah Theater

Gambar 4.3.28 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk menambah theater. Terdapat kode program untuk membuat modal dengan judul “Tambah Bioskop” di mana isi dari modal tersebut terdapat *input text*, *select*, tombol “Batal”, dan tombol “Simpan”. Untuk *input text* akan memunculkan *placeholder* “Nama Bioskop”, di mana *input text* berfungsi untuk menerima input Nama Bioskop. Untuk menghindari *input* data kosong, maka *input text* ini dibuat menjadi *required*. Untuk *select* menggunakan fitur dari Select2. Isi dari *select* ini adalah data-data dari const KOTA yang terdapat pada model Theater. Untuk menghindari *input* data kosong, maka *select* ini dibuat menjadi *required*. Tombol “Batal” untuk membatalkan penambahan theater. Tombol “Simpan” untuk menambah/menyimpan data theater.

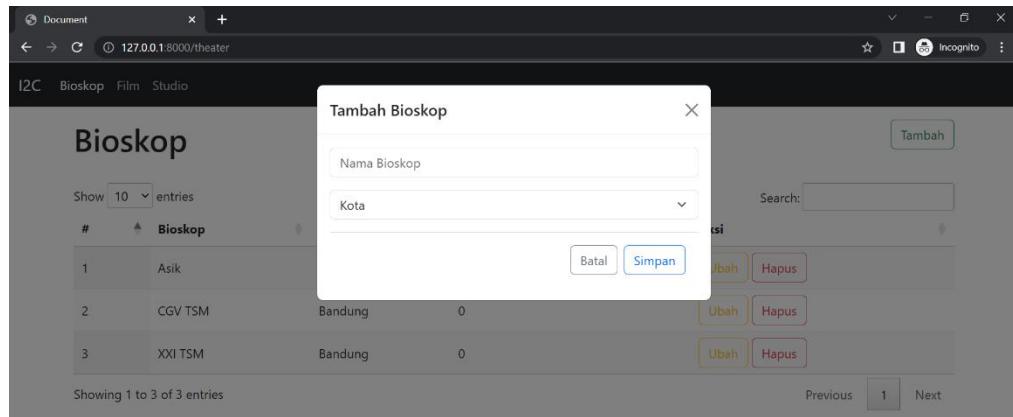
```

22     public function store_theater(Request $request){
23         $datanamabioskop = $request->nama_bioskop;
24         $datakota = $request->kota;
25         $datastudio = 0;
26         $theaters = new Theater();
27         $theaters->nama_bioskop=$datanamabioskop;
28         $theaters->kota=$datakota;
29         $theaters->jumlah_studio=$datastudio;
30         try {
31             $theaters->save();
32         } catch (QueryException $e) {
33
34             return back()->withErrors(["status"=>"Fail to input data"]);
35         }
36         return redirect("/theater");
37     }

```

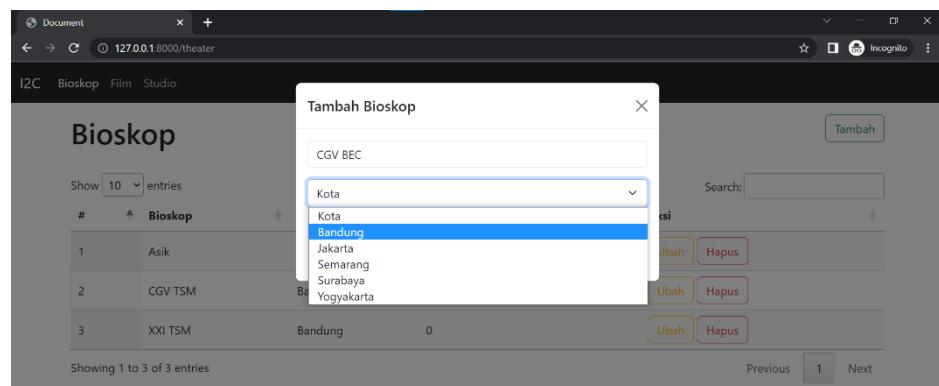
Gambar 4.3. 29 kode program controller create Theater

Gambar 4.3.29 adalah kode program yang penulis kerjakan pada *controller* untuk menambahkan data theater. Nama fungsinya adalah “store_theater”. Pada fungsi ini, data-data yang diinput dari modal akan ditambahkan ke dalam tabel theaters.



Gambar 4.3. 30 tampilan modal tambah Theater

Gambar 4.3.30 adalah tampilan modal untuk menambahkan data theater. Tampilan modal ini akan muncul setelah tombol “Tambah” diklik. Modal tersebut menggunakan fitur dari Bootstrap.



Gambar 4.3. 31 tampilan select dengan Select2

Gambar 4.3.31 adalah tampilan dari penerapan Select2. Ketika penulis mengetik huruf “b” pada select, select tersebut langsung meng-highlight “Bandung”. Hal ini merupakan keuntungan dari menggunakan Select2. Pada kasus ini, penulis ingin menambahkan data baru, yaitu “CGV BEC” sebagai Nama Bioskop dan “Bandung” sebagai Kota. Kemudian klik “Simpan” untuk menyimpan atau menambah data theater baru tersebut.

#	Bioskop	Kota	Jumlah Studio	Aksi
1	Asik	Bandung	0	<button>Ubah</button> <button>Hapus</button>
2	CGV BEC	Bandung	0	<button>Ubah</button> <button>Hapus</button>
3	CGV TSM	Bandung	0	<button>Ubah</button> <button>Hapus</button>
4	XXI TSM	Bandung	0	<button>Ubah</button> <button>Hapus</button>

Gambar 4.3.32 tampilan data theater berhasil ditambah

Gambar 4.3.32 adalah tampilan ketika tombol “Simpan” pada modal diklik dan data theater baru berhasil ditambahkan ke dalam tabel theaters. Terlihat bahwa data *theater* baru tersebut langsung ditampilkan pada tabel.

4.3.3.3 Update

Update Theater

- Munculkan Modal "Edit Bioskop" yg berisikan form dgn fields sbb:
 - **Catatan: Terapkan LaravelCollective pada penggunaan Form.**
 - Input hidden dgn value **data-th->id**.
 - **data-th** diperoleh dari atribut yg terdapat di setiap tombol "Edit". Manfaatkan JQuery.
 - Input text "Edit Nama Bioskop".
 - Default value = **data-th->nama_bioskop**.
 - Required.
 - Select "Edit Kota".
 - Data / item diisi oleh const KOTA dari Model Theater.
 - Value = Teks = item dari const KOTA.
 - Placeholder: "Pilih Kota".
 - Default selected = **data-th->kota**.
 - Required.
 - Pada Modal Footer:
 - Tombol "Batal". Ketika di-klik, CLOSE Modal.
 - Tombol "Perbarui". Ketika di-klik, UPDATE data di database.
- UPDATE data di database:
 - Route: /theater
 - Controller: TheaterController@update_theater
 - Method: PATCH
 - Rincian di Controller:
 - **Step 1**
 - FIND data pada tabel theaters yg memiliki id = value input hidden.

- Jika ADA data, maka lanjut ke **Step 2**.
- **Step 2**
 - Update data di database:
 - **nama_bioskop** = value input "Edit Nama Bioskop".
 - **kota** = value select "Edit Kota" yg dipilih.
- **Step 3**
 - Return redirect ke halaman /theater.

Gambar 4.3. 33 task update Theater

Gambar 4.3.33 adalah *task* untuk membuat bagian *update* pada modul Theater.

```
22     Route::PATCH('/theater', [TheaterController::class, "update_theater"]);
```

Gambar 4.3. 34 kode program route update Theater

Gambar 4.2.34 adalah kode program yang telah penulis kerjakan untuk membuat *route update* untuk mengubah data modul Theater. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *patch*,

karena berfungsi untuk mengubah data. Untuk mengubah data, maka fungsi yang akan dipanggil dalam *controller* bernama “update_theater”.

```

79 | <div class="modal fade" tabindex="-1" id="updatebioskop">
80 |   <div class="modal-dialog">
81 |     <div class="modal-content">
82 |       <div class="modal-header">
83 |         <h5 class="modal-title" style="margin: 0;">Edit Bioskop

```

Gambar 4.3. 35 kode program tampilan Theater

Gambar 4.3.35 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk mengubah data theater. Terdapat kode program untuk membuat modal dengan judul “Edit Bioskop” di mana isi dari modal tersebut terdapat *input text*, *select*, tombol “Batal”, dan tombol “Perbarui”. Untuk *input text* akan memunculkan Nama Bioskop tersebut. Nama Bioskop tersebut dapat diubah atau dibiarkan saja. Untuk *select* menggunakan fitur dari Select2. Isi dari *select* ini adalah data-data dari *const KOTA* yang terdapat pada model *Theater*. Select ini akan langsung berisi kota dari *theater* tersebut. Tombol “Batal” untuk membatalkan pengubahan *theater*. Tombol “Perbarui” untuk memperbarui atau mengubah data theater.

```

109 | <script>
110 |   function editFunction() {
111 |     const th = $(this).data('th');
112 |     $('#id').val(th.id);
113 |     $('#nama_bioskop').val([th.nama_bioskop])
114 |     $('#kota').val(th.kota)
115 |   }
116 |   $(".btn-edit").click(editFunction);

```

Gambar 4.3. 36 kode program script edit Theater

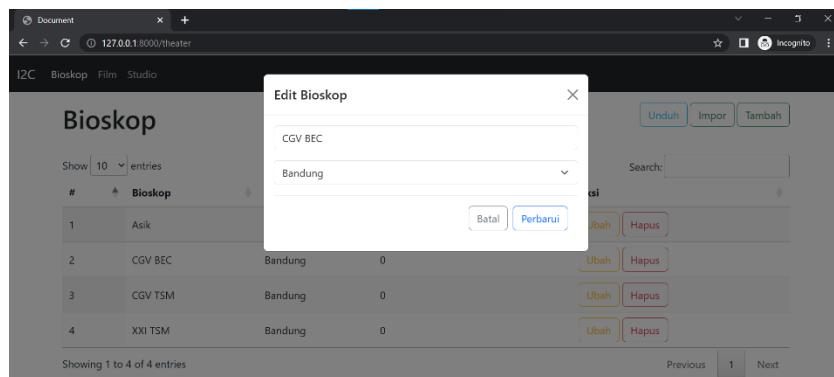
Gambar 4.3.36 adalah kode program pada bagian script agar data Nama Bioskop dan Kota dari data *theater* yang ingin diubah dapat ditampilkan pada bagian modal “Edit Bioskop”.

```

39     public function update_theater(Request $request){
40         $theater = Theater::find($request->id);
41         $datanamabioskop = $request->edit_nama_bioskop;
42         $datakota = $request->edit_kota;
43         $theater->nama_bioskop=$datanamabioskop;
44         $theater->kota=$datakota;
45         try {
46             $theater->update();
47         } catch (\Throwable $th) {
48             return back()->withErrors(["status"=>"Fail to update data"]);
49         }
50     }
51     return redirect("/theater");
52 }
```

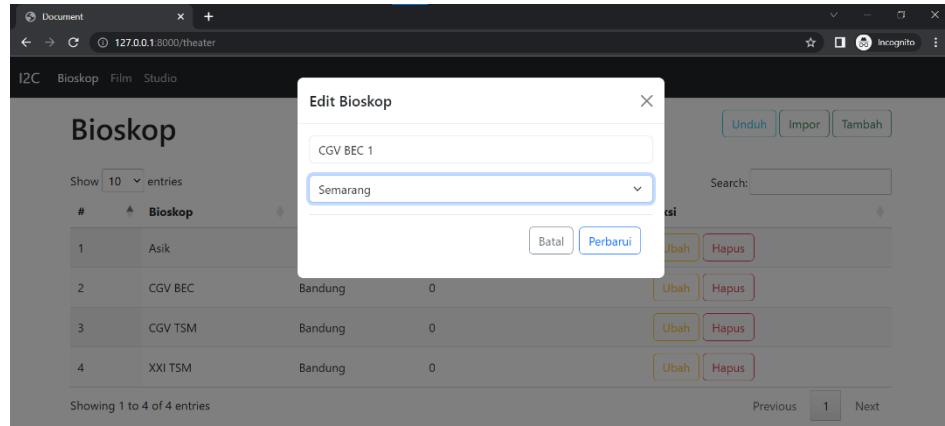
Gambar 4.3.37 kode program *controller* ubah Theater

Gambar 4.3.37 adalah kode program yang penulis kerjakan pada *controller* untuk mengubah data theater. Nama fungsinya adalah “*update_theater*”. Pada fungsi ini, pertama-tama sistem akan mencari / *find* id yang sesuai dengan data theater tersebut. Jika id sudah ditemukan, selanjutnya Nama Bioskop dan Kota dari id *theater* tersebut akan diubah sesuai dengan yang di-*input* pada modal.



Gambar 4.3.38 tampilan modal ubah Theater

Gambar 4.3.38 adalah tampilan ketika tombol “Ubah” pada data *theater* baru diklik. Ketika diklik, akan muncul modal untuk mengubah data *theater*. Pada bagian *input text* dan *select* langsung terisi Nama Bioskop yaitu “CGV BEC” dan Kota yaitu “Bandung”.



Gambar 4.3. 39 tampilan ingin mengubah Theater

Gambar 4.3.39 adalah tampilan ketika penulis ingin mengubah Nama Bioskop menjadi “CGV BEC 1” dan Kota menjadi “Semarang”. Kemudian klik tombol “Perbarui” untuk mengubah data *theater* tersebut.

#	Bioskop	Kota	Jumlah Studio	Aksi
1	Asik	Bandung	0	<button>Ubah</button> <button>Hapus</button>
2	CGV BEC 1	Semarang	0	<button>Ubah</button> <button>Hapus</button>
3	CGV TSM	Bandung	0	<button>Ubah</button> <button>Hapus</button>
4	XXI TSM	Bandung	0	<button>Ubah</button> <button>Hapus</button>

Gambar 4.3. 40 tampilan Theater berhasil diubah

Gambar 4.3.40 adalah tampilan ketika tombol “Perbarui” pada modal diklik dan data *theater* berhasil diubah. Terlihat bahwa data theater yang baru diubah tersebut langsung ditampilkan pada tabel.

4.3.3.4 Delete

Delete Theater

- Munculkan SWAL konfirmasi dgn rincian sbb:
 - SWAL = Sweet Alert. Pelajari di sini utk penerapannya:
 - Instalasi menggunakan script, **Jangan** npm install: <https://sweetalert2.github.io/#download>.
 - ```
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11">
</script>
```
  - Contoh Pemakaian SWAL Konfirmasi:
 

```
swal.fire({
 text: "Apakah Anda yakin ingin menghapus data ini?",
 icon: 'warning',
 showCancelButton: true,
 confirmButtonColor: '#3085d6',
 cancelButtonColor: '#d33',
 confirmButtonText: 'Delete'
}).then((result) => {
 if (result.isConfirmed) {
 $('#delete_kc_form').submit();
 }
});
```
- Tulisan: "Yakin untuk menghapus Bioskop ini?"
  - Tombol "Batal". Close SWAL.
  - Tombol "Hapus". Ketika di-klik, maka DELETE data di database.
- DELETE data di database:
  - Route: /theater
  - Controller: TheaterController@delete\_theater
  - Method: DELETE
  - Rincian di Controller:
    - **Step 1**
      - FIND data pada tabel theaters yg memiliki id = **data-id** dari item yg dipilih. **data-id** diperoleh dari atribut pada tombol. Manfaatkan JQuery.
      - Jika ADA data, maka lanjut ke **Step 2**.

- 
- **Step 2**
    - Delete data ybs di database.
  - **Step 3**
    - Return redirect ke halaman /theater.

**Gambar 4.3. 41 task delete Theater**

Gambar 4.3.41 adalah *task* untuk membuat bagian *delete* pada modul Theater.

```
23 Route::DELETE('/theater', [TheaterController::class, "delete_theater"]);
```

**Gambar 4.3. 42 kode program route delete Theater**

Gambar 4.3.42 adalah kode program yang telah penulis kerjakan untuk membuat *route delete* untuk menghapus data modul Theater. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *delete*, karena berfungsi untuk menghapus data. Untuk menghapus data, maka fungsi yang akan dipanggil dalam *controller* bernama “*delete\_theater*”.

```

104 <div>
105 {{ Form::open(['method' => 'DELETE', 'id' => 'deleteform']) }}
106 {{ Form::hidden('id', null, ['id' => 'deleteid']) }}
107 {{ Form::close() }}
108 </div>
109 <script>
110 $('.btn-delete').click(function() {
111 Swal.fire({
112 icon: "warning",
113 text: "Yakin untuk menghapus Bioskop ini?",
114 showCancelButton: true,
115 confirmButtonColor: '#3085d6',
116 cancelButtonColor: '#d33',
117 cancelButtonText: 'Batal',
118 confirmButtonText: 'Hapus'
119 }).then((result) => {
120 if (result.isConfirmed) {
121 const id = $(this).data("id");
122 console.log(id);
123 $('#deleteid').val(id);
124 $('#deleteform').submit();
125 }
126 })
127 })
128 </script>

```

Gambar 4.3. 43 kode program tampilan hapus Theater

Gambar 4.3.43 kode program yang penulis kerjakan pada *view* untuk membuat tampilan ketika ingin menghapus data theater yang dipilih. Untuk tampilannya menggunakan fitur dari SWAL (*Sweet Alert*). Terdapat kode program untuk membuat form *delete*. Kemudian terdapat kode program untuk penggunaan SWAL. Pada SWAL ini terdapat pesan “Yakin untuk menghapus Bioskop ini?”, tombol “Hapus” untuk menghapus *theater* tersebut, serta tombol “Kembali” untuk kembali ke halaman *main page*.

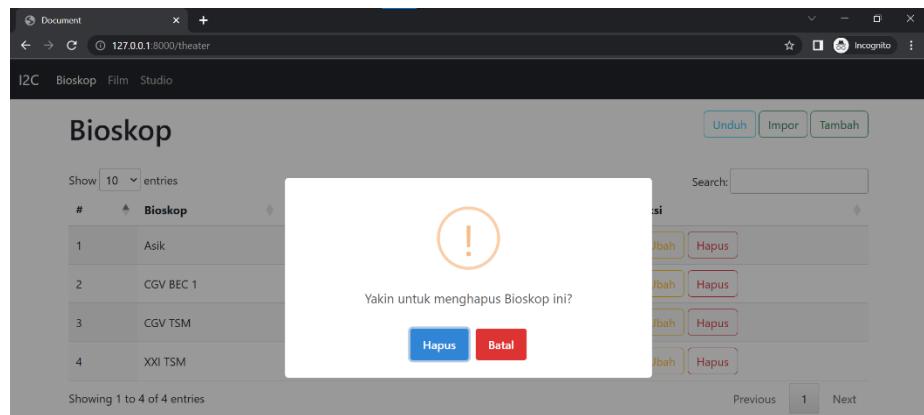
```

53 public function delete_theater(Request $request){
54 $id = $request->id;
55 $theater = Theater::find($id);
56 if($theater != null){
57 $result = $theater->delete();
58 }
59 if($result){
60 return redirect("/theater");
61 }else{
62 return back()->withErrors("Fail to Delete");
63 }
64 }

```

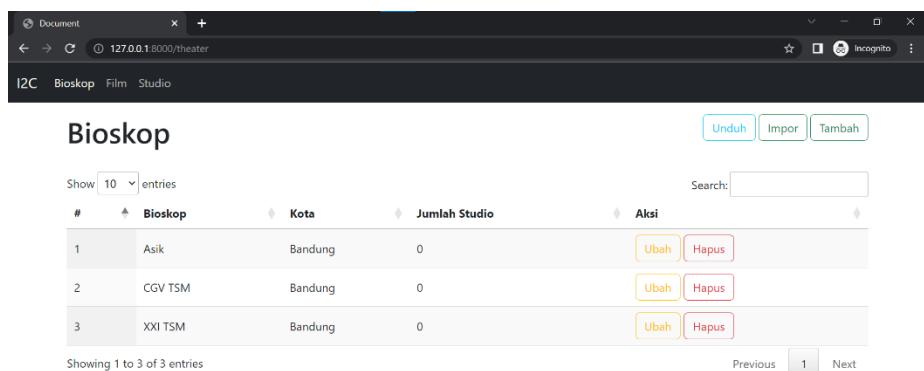
Gambar 4.3. 44 kode program controller hapus Theater

Gambar 4.3.44 adalah kode program yang penulis kerjakan pada *controller* untuk menghapus data theater. Nama fungsinya adalah “*delete\_theater*”. Pada fungsi ini, pertama-tama sistem akan mencari/*find* id yang sesuai dengan data *theater* tersebut. Jika id sudah ditemukan, selanjutnya data theater tersebut akan dihapus dari tabel theaters.



Gambar 4.3. 45 tampilan SWAL hapus Theater

Gambar 4.3.45 adalah tampilan ketika tombol “Hapus” dari data theater yang baru diubah diklik. Ketika diklik, maka akan muncul tampilan SWAL yang memunculkan pesan konfirmasi terlebih dahulu sebelum data *theater* tersebut dihapus. Untuk menghapus data tersebut, selanjutnya klik tombol “Hapus”.



Gambar 4.3. 46 tampilan Theater berhasil dihapus

Gambar 4.3.46 adalah tampilan ketika tombol “Hapus” pada SWAL diklik. Ketika diklik, data *theater* tersebut dihapus dan tidak muncul kembali pada tabel theaters.

#### 4.3.4 CRUD Film

Untuk modul Film pertama-tama penulis membuat untuk bagian *read* dahulu, kemudian *create*, *update*, dan terakhir *delete*.

##### 4.3.4.1 Read

###### Modul Film

###### Main Page

- Route: /film
- Controller: FilmController@index\_film
- Method: GET
- View: views/film.blade.php.
- Tampilan:
  - Tampilan dibuat seperti pada Modul Theater.
  - Tulisan Judul: "Film".
  - Tombol di sebelah kanan atas: Tombol "Tambah". Ketika di-klik, munculkan Modal "Tambah Film". Lihat bagian "**Insert Film**".
  - Rincian Tabel:
    - GET data dari tabel films, order by film ASCENDING.

| # | Judul | Durasi | Rating | Sinopsis | Rilis | Bahasa | Aksi |
|---|-------|--------|--------|----------|-------|--------|------|
|   |       |        |        |          |       |        |      |
|   |       |        |        |          |       |        |      |
|   |       |        |        |          |       |        |      |
|   |       |        |        |          |       |        |      |

- Kolumn "#" => diisi oleh urutan mulai dari 1 s/d jumlah data.
- Kolumn "Judul" => films->film.
- Kolumn "Durasi" => "{films->durasi} menit".
- Kolumn "Rating" => films->rating.
- Kolumn "Sinopsis":
  - Jika films->sinopsis = null, maka tampilkan "-".
  - Jika films->sinopsis TIDAK null, maka tampilkan "(films->sinopsis)".
- Kolumn "Rilis" => films\_details->jadwal\_rilis.
- Kolumn "Bahasa" => films\_details->bahasa.
- Kolumn "Aksi":
  - Tombol "Edit".
    - Sertakan atribut **data-f** = {films dari data ybs}.
    - Sertakan atribut **data-fd** = {films\_details dari data ybs}.
    - Ketika tombol di-klik, munculkan Modal "Edit Film". Lihat bagian "**Update Film**".
  - Tombol "Hapus".

- Sertakan atribut **data-id** = {films->id dari data ybs}.
- Lihat bagian "**Delete Film**".

Gambar 4.3.47 task main page / read Film

Gambar 4.3.47 adalah *task* untuk membuat *main page* untuk modul Film.

```
25 Route::GET('/film', [FilmController::class, "index_film"]);
```

Gambar 4.3.48 kode program route read Film

Gambar 4.3.48 adalah kode program yang telah penulis kerjakan untuk membuat *route read* untuk menampilkan data modul Film. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *get*, karena berfungsi untuk menampilkan data. Untuk menampilkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “index\_film”.

```

1 @extends('master')
2 @section('content')
3
4

5

6

7 @if ($errors->any())
8 <div class="alert alert-danger">
9 {{ $errors->first() }}
10 </div>
11 @endif
12 <div class="d-flex justify-content-between">
13 <h1>Film</h1>
14 <div>
15 <button data-bs-target="#" data-bs-toggle="modal" class="btn btn-outline-info">Unduh</button>
16 <button data-bs-target="#" data-bs-toggle="modal" class="btn btn-outline-success">Import</button>
17 <button data-bs-target="#tambahfile" data-bs-toggle="modal" class="btn btn-outline-success">Tambah</button>
18 </div>
19

20
21 <table id="table_id" class="display">
22 <thead>
23 <th>#</th>
24 <th>Judul</th>
25 <th>Durasi</th>
26 <th>Rating</th>
27 <th>Sinopsis</th>
28 <th>Rilis</th>
29 <th>Bahasa</th>
30 <th>Aksi</th>
31
32 </thead>
33 <tbody>
34 @foreach ($films as $item)
35 <tr>
36 <td>
37 {{ $loop->iteration }}
38 </td>
39 <td>
40 {{ $item->film }}
41 </td>
42 <td>
43 {{ $item->durasi }} menit
44 </td>
45 <td>
46 {{ $item->rating }}
47 </td>
48 <td>
49 @if ($item->sinopsis != null)
50 {{ $item->sinopsis }}
51 @else
52 -
53 @endif
54
55 </td>
56 <td>
57 {{ $item->film_filmdetail->jadwal_rilis }}
58 </td>
59 <td>
60 {{ $item->film_filmdetail->bahasa }}
61 </td>
62 <td>
63 <button class="btn btn-outline-warning btn-edit" data-f="{{ $item }}"
64 data-fd="{{ $item->film_filmdetail }}"
65 data-bs-toggle="modal">Edit</button>
66 <button class="btn btn-outline-danger btn-delete" data-id="{{ $item->id }}>Hapus</button>
67 </td>
68 </tr>
69 @endforeach
70 </tbody>
71 </table>

```

Gambar 4.3. 49 kode program tampilan Film

Gambar 4.3.49 adalah kode program yang penulis kerjakan untuk membuat tampilan dari modul Film. Pada tampilan ini akan menampilkan tabel yang

menggunakan DataTables. Tabel ini akan menampilkan data-data dari tabel films dan tabel films\_details. Terdapat juga tombol Ubah untuk mengubah data film tersebut dan Hapus untuk menghapus data film tersebut.

```

12 public function index_film()
13 {
14 $films = Film::orderBy("film","ASC")->get();
15 $ratingcoll = [];
16 foreach(Film::RATING as $item){
17 $ratingcoll[$item]=$item;
18 }
19 $bahasacoll = [];
20 foreach(FilmDetail::BAHASA as $item){
21 $bahasacoll[$item]=$item;
22 }
23 return view("film",["films"=>$films,"ratingcoll"=>$ratingcoll,"bahasacoll"=>$bahasacoll]);
24 }

```

Gambar 4.3. 50 kode program *controller* Film

Gambar 4.3.50 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data dari tabel films. Data tersebut diurutkan berdasarkan abjad dari kolom film secara *ascending* atau menurun.

Tabel films dan films\_details saat ini masih kosong atau belum ada data, sehingga penulis perlu mengisi data *dummy* ke dalam tabel-tabel tersebut. Untuk mengisi data *dummy* ke dalam tabel theaters, penulis menggunakan salah satu fitur dari Laravel yaitu *seeder*.

```

database > seeders > ** FilmSeeder.php
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class FilmSeeder extends Seeder
10 {
11 /**
12 * Run the database seeds.
13 */
14 public function run(): void
15 [
16 /**
17 * DB::table('films')->insert([
18 "film"=>"Film1",
19 "durasi"=>3,
20 "rating"=>"SU",
21 "sinopsis"=>"Film1 keren",
22]);
23]
24 }
25

```

Gambar 4.3. 51 kode program *seeder* Film

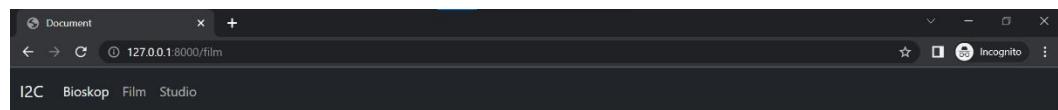
Gambar 4.3.51 adalah kode program yang telah penulis kerjakan pada bagian *seeder* films. Kolom pada tabel films yang perlu diisi data *dummy* adalah kolom film, kolom durasi, kolom rating, dan kolom sinopsis.

```

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class FilmDetailSeeder extends Seeder
10 {
11 /**
12 * Run the database seeds.
13 */
14 public function run(): void
15 {
16 DB::table('films_details')->insert([
17 "film_id"=>"1",
18 "jadwal_rilis"=>"3/2/2023",
19 "bahasa"=>"Indonesia",
20],
21);
22 }
23 }
24 }
```

**Gambar 4.3. 52 kode program *seeder* tabel  
films\_details**

Gambar 4.3.52 adalah kode program yang telah penulis kerjakan pada bagian *seeder* films\_details. Kolom pada tabel films\_details yang perlu diisi data *dummy* adalah kolom film\_id, kolom jadwal rilis, dan kolom bahasa.



| # | Judul | Durasi  | Rating | Sinopsis    | Rilis      | Bahasa    | Aksi                                         |
|---|-------|---------|--------|-------------|------------|-----------|----------------------------------------------|
| 1 | Film1 | 3 menit | SU     | Film1 keren | 2023-03-02 | Indonesia | <button>Edit</button> <button>Hapus</button> |

**Gambar 4.3. 53 tampilan modul Film**

Gambar 4.3.53 adalah tampilan dari modul Film. Tampilan ini memiliki judul “Film”. Lalu terdapat tabel yang berisi kolom untuk penomoran, Judul, Durasi, Rating, Sinopsis, Rilis, Bahasa, serta Aksi di mana kolom ini berisi tombol Ubah dan Hapus. Selain itu terdapat juga tombol Tambah untuk menampilkan modal untuk menambah data Film.

#### 4.3.4.2 Create

##### Insert Film

- Munculkan Modal "Tambah Film" yg berisikan form dgn fields sbb:
  - **Catatan: Terapkan LaravelCollective pada penggunaan Form.**
  - Input text "Judul Film". Required. Unique.
  - Input number "Durasi". Required.
  - Select "Rating".
    - Data / item diisi oleh const RATING dari Model Film.
    - Value = Teks = item dari const RATING.
    - Placeholder: "Pilih Rating".
    - Required.
  - TextArea "Sinopsis". Tidak Required.
  - Input date "Jadwal Rilis". Required.
  - Select "Bahasa":
    - Data / item diisi oleh const BAHASA dari Model FilmDetail.
    - Value = Teks = item dari const BAHASA.
    - Placeholder: "Pilih Bahasa".
    - Required.
  - Pada Modal Footer:
    - Tombol "Batal". Ketika di-klik, CLOSE Modal.
    - Tombol "Simpan". Ketika di-klik, INSERT data ke database.
- INSERT data ke database:
  - Route: /film
  - Controller: FilmController@store\_film
  - Method: POST
  - Rincian di Controller:
    - **Step 1**
      - Insert data ke tabel films:
        - film = value input "Judul Film".
        - durasi = value input "Durasi".
        - rating = value select "Rating".
        - sinopsis = value textarea "Sinopsis".
    - **Step 2**
      - Insert data ke tabel films\_details:
        - film\_id = films->id yg telah di-insert di **Step 1**.
        - jadwal\_rilis = value input "Jadwal Rilis".
        - bahasa = value select "Bahasa".
    - **Step 3**
      - Return redirect ke halaman /film.

**Gambar 4.3. 54 task create Film**

Gambar 4.3.54 adalah *task* untuk membuat bagian *create* atau *insert* pada modul Film.

```
26 Route::POST('/film', [FilmController::class, "store_film"]);
```

**Gambar 4.3. 55 kode program route create Film**

Gambar 4.3.55 adalah kode program yang telah penulis kerjakan untuk membuat *route create* untuk menambahkan data modul Film. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *post*, karena berfungsi untuk menambahkan data. Untuk menambahkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “*store\_film*”.

```

72 <div class="modal fade" tabindex="-1" id="tambahfilm">
73 <div class="modal-dialog">
74 <div class="modal-content">
75 <div class="modal-header">
76 <h5 class="modal-title">Tambah Film</h5>
77 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
78 </div>
79 <div class="modal-body">
80 {{ Form::open(['method' => 'POST', 'url' => '/film']) }}
81 <div class="mb-3">
82 {{ Form::text('judul_film', null, ['class' => 'form-control', 'placeholder' => 'Judul Film', 'required', 'unique']) }}
83 </div>
84 <div class="mb-3">
85 {{ Form::number('durasi', null, ['class' => 'form-control', 'placeholder' => 'Durasi', 'required']) }}
86 </div>
87 <div class="mb-3">
88 {{ Form::select('rating', $ratingcoll, null, ['class' => 'form-select', 'placeholder' => 'Rating', 'required']) }}
89 </div>
90 <div class="mb-3">
91 {{ Form::textarea('sinopsis', null, ['class' => 'form-control', 'placeholder' => 'Sinopsis']) }}
92 </div>
93 <div class="mb-3">
94 {{ Form::date('jadwal_rilis', null, ['class' => 'form-control', 'placeholder' => 'Jadwal Rilis', 'required']) }}
95 </div>
96 <div class="mb-3">
97 {{ Form::select('bahasa', $bahasacoll, null, ['class' => 'form-select', 'placeholder' => 'Bahasa', 'required']) }}
98 </div>
99 <div class="modal-footer">
100 {{ Form::reset('Batal', ['class' => 'btn btn-outline-secondary', 'data-bs-dismiss' => 'modal']) }}
101 {{ Form::submit('Simpan', ['class' => 'btn btn-outline-primary']) }}
102 </div>
103 {{ Form::close() }}
104 </div>
105 </div>
106 </div>
107 </div>

```

Gambar 4.3. 56 kode program tampilan modal tambah Film

Gambar 4.3.56 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk menambah film. Terdapat kode program untuk membuat modal dengan judul “Tambah Film” di mana isi dari modal tersebut terdapat *input text*, *number*, *select* pertama, *textarea*, *date*, *select* kedua, tombol “Batal”, dan tombol “Simpan”. Untuk *input text* akan memunculkan *placeholder* “Judul Film”, di mana *input text* ini berfungsi untuk menerima input Judul Film. Untuk menghindari *input* data kosong, maka *input text* ini dibuat menjadi *required*. Lalu *input number* ini berfungsi untuk menerima input Durasi. Untuk menghindari *input* data kosong, maka *input number* ini dibuat menjadi *required*. Untuk *select* pertama menggunakan fitur dari Select2. Isi dari *select* ini adalah data-data dari const RATING yang terdapat pada model Film. Untuk menghindari *input* data kosong, maka *select* ini dibuat menjadi *required*. Untuk *input textarea* akan memunculkan *placeholder* “Sinopsis”, di mana *input textarea* ini berfungsi untuk menerima input Sinopsis. Untuk *input date* akan memunculkan *placeholder* “Jadwal Rilis”, di mana *input date* ini berfungsi untuk menerima input Jadwal Rilis. Untuk menghindari *input* data kosong, maka *input date* ini dibuat menjadi *required*. Untuk *select* kedua menggunakan fitur dari Select2. Isi dari *select* ini adalah data-data dari const BAHASA yang terdapat pada model FilmDetail. Untuk menghindari *input* data kosong, maka *select* ini dibuat menjadi *required*. Tombol

“Batal” untuk membatalkan penambahan film. Tombol “Simpan” untuk menambah/menyimpan data film.

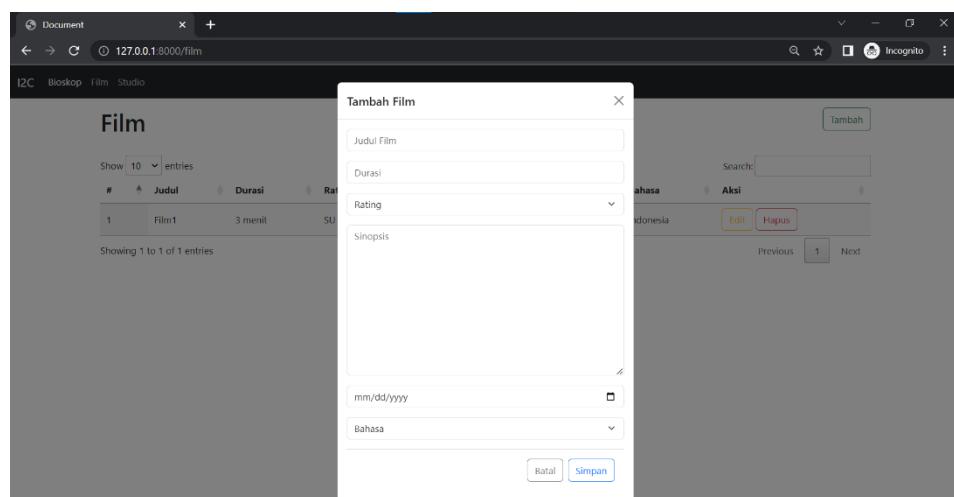
```

26 | public function store_film(Request $request){
27 | $validation = $request->validate([
28 | "judul_film"=>"required | unique:films,film"
29 |]);
30 | $datafilm = $validation["judul_film"];
31 | $datadurasi = $request->durasi;
32 | $datarating = $request->rating;
33 | $datasinopsis = $request->sinopsis;
34 | $datajadwalrilis = $request->jadwal_rilis;
35 | $databahasa = $request->bahasa;
36 | $films = new Film();
37 | $details = new FilmDetail();
38 | $films->film=$datafilm;
39 | $films->durasi=$datadurasi;
40 | $films->rating=$datarating;
41 | $films->sinopsis=$datasinopsis;
42 | try {
43 | $films->save();
44 | } catch (QueryException $e) {
45 |
46 | return back()->withErrors(["status"=>"Fail to input data"]);
47 | }
48 | $details->film_id=$films->id;
49 | $details->jadwal_rilis=$datajadwalrilis;
50 | $details->bahasa=$databahasa;
51 | try {
52 | $details->save();
53 | } catch (QueryException $e) {
54 | $films->forceDelete();
55 | return back()->withErrors(["status"=>"Fail to input data"]);
56 | }
57 | return redirect("/film");
58 | }

```

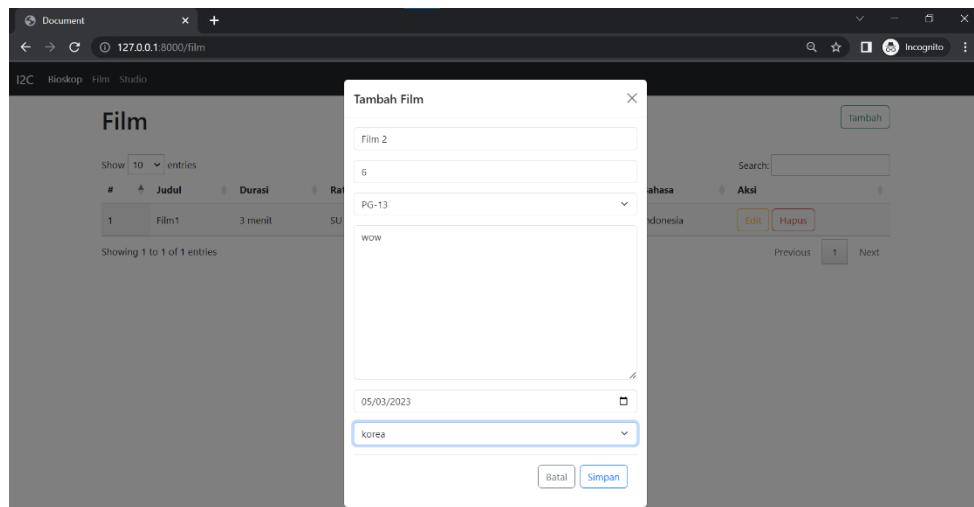
Gambar 4.3. 57 kode program *controller tambah Film*

Gambar 4.3.57 adalah kode program yang penulis kerjakan pada *controller* untuk menambahkan data film. Nama fungsinya adalah “store\_film”. Pada fungsi ini, data-data yang diinput dari modal akan ditambahkan ke dalam tabel films.



Gambar 4.3. 58 tampilan modal tambah Film

Gambar 4.3.58 adalah tampilan modal untuk menambahkan data film. Tampilan modal ini akan muncul setelah tombol “Tambah” diklik. Modal tersebut menggunakan fitur dari Bootstrap.



Gambar 4.3. 59 tampilan menambah Film

Gambar 4.3.59 adalah tampilan ketika penulis ingin menambahkan data baru, yaitu “Film2” sebagai Judul Film, “6” sebagai Durasi, “PG-13” sebagai Rating, “wow” sebagai Sinopsis, “05/03/2023” sebagai Jadwal Rilis, dan “Korea” sebagai Bahasa. Kemudian klik “Simpan” untuk menyimpan atau menambah data film baru tersebut.

| # | Judul | Durasi  | Rating | Sinopsis    | Rilis      | Bahasa    | Aksi                                         |
|---|-------|---------|--------|-------------|------------|-----------|----------------------------------------------|
| 1 | Film1 | 3 menit | SU     | Film1 keren | 2023-03-02 | Indonesia | <button>Edit</button> <button>Hapus</button> |
| 2 | Film2 | 6 menit | PG-13  | wow         | 2023-05-03 | korea     | <button>Edit</button> <button>Hapus</button> |

Gambar 4.3. 60 tampilan Film berhasil ditambah

Gambar 4.3.60 adalah tampilan ketika tombol “Simpan” pada modal diklik dan data film baru berhasil ditambahkan ke dalam tabel films. Terlihat bahwa data theater baru tersebut langsung ditampilkan pada tabel.

### 4.3.4.3 Update

#### Update Film

- Munculkan Modal "Edit Film" yg berisikan form dgn fields sbb:
  - **Catatan:** Terapkan LaravelCollective pada penggunaan Form.
  - Input hidden dgn value = **data-f->id**.
  - Input text "Edit judul Film".
    - Default Value = **data-f->film**.
    - Required.
    - Unique.
  - Input number "Edit Durasi".
    - Default value = **data-f->durasi**.
    - Required.
  - Select "Edit Rating".
    - Data / item diisi oleh const RATING dari Model Film.
    - Value = Teks = item dari const RATING.
    - Placeholder: "Pilih Rating".
    - Default selected = **data-f->rating**.
    - Required.
  - TextArea "Edit Sinopsis".
    - Default value = **data-f->sinopsis**.
    - Tidak Required.
  - Input date "Edit Jadwal Rilis".
    - Default value = **data-fd->jadwal\_rilis**.
    - Required.
  - Select "Edit Bahasa".
    - Data / item diisi oleh const BAHASA dari Model FilmDetail.
    - Value = Teks = item dari const BAHASA.
    - Placeholder: "Pilih Bahasa".
    - Default selected = **data-fd->bahasa**.
    - Required.
  - Pada Modal Footer:
    - Tombol "Batal". Ketika di-klik, CLOSE Modal.
    - Tombol "Perbarui". Ketika di-klik, UPDATE data ke database.
- UPDATE data ke database:
  - Route: /film
  - Controller: FilmController@update\_film
  - Method: PATCH
  - Rincian di Controller:
    - **Step 1**
      - FIND data dari tabel films yg memiliki id = value input hidden.
      - jika ADA data, maka lanjut ke **Step 2**.
    - **Step 2**
      - Update data ybs di tabel films:
        - film = value input "Edit Judul Film".
        - durasi = value input "Edit Durasi".
    - **Step 3**
      - Update data yg memiliki film\_id = films->id di tabel films\_details:
        - jadwal\_rilis = value input "Edit Jadwal Rilis".
        - bahasa = value select "Edit Bahasa".
    - **Step 4**
      - Return redirect ke halaman /film.

Gambar 4.3. 61 task update Film

Gambar 4.3.61 adalah *task* untuk membuat bagian *update* pada modul Film.

```
27 Route::PATCH('/film', [FilmController::class, "update_film"]);
```

Gambar 4.3. 62 kode program *route update* Film

Gambar 4.3.62 adalah kode program yang telah penulis kerjakan untuk membuat *route update* untuk mengubah data modul Film. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *patch*, karena berfungsi untuk mengubah data. Untuk mengubah data, maka fungsi yang akan dipanggil dalam *controller* bernama “*update\_film*”.

```

108 <div class="modal fade" tabindex="-1" id="updatefilm">
109 <div class="modal-dialog">
110 <div class="modal-content">
111 <div class="modal-header">
112 <h5 class="modal-title">Edit Film</h5>
113 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
114 </div>
115 <div class="modal-body">
116 {{ Form::open(['method' => 'PATCH']) }}
117 {{ Form::hidden('id', null, ['id' => 'id']) }}
118 <div class="mb-3">
119 {{ Form::text('edit_judul_film', null, ['class' => 'form-control', 'placeholder' => 'Edit Judul Film', 'required', 'unique', 'id' => 'film']) }}
120 </div>
121 <div class="mb-3">
122 {{ Form::number('edit_durasi', null, ['class' => 'form-control', 'placeholder' => 'Edit Durasi', 'required', 'id' => 'durasi']) }}
123 </div>
124 <div class="mb-3">
125 {{ Form::select('edit_rating', $ratingcoll, null, ['class' => 'form-select', 'placeholder' => 'Edit Rating', 'required', 'id' => 'rating']) }}
126 </div>
127 <div class="mb-3">
128 {{ Form::textarea('edit_sinopsis', null, ['class' => 'form-control', 'placeholder' => 'Edit Sinopsis', 'id' => 'sinopsis']) }}
129 </div>
130 <div class="mb-3">
131 {{ Form::date('edit_jadwal_rilis', null, ['class' => 'form-control', 'placeholder' => 'Edit Jadwal Rilis', 'required', 'id' => 'jadwal_rilis']) }}
132 </div>
133 <div class="mb-3">
134 {{ Form::select('edit_bahasa', $bahasacoll, null, ['class' => 'form-select', 'placeholder' => 'Pilih Bahasa', 'required', 'id' => 'bahasa']) }}
135 </div>
136 <div class="modal-footer">
137 {{ Form::reset('Batal', ['class' => 'btn btn-outline-secondary', 'data-bs-dismiss' => 'modal']) }}
138 {{ Form::submit('Perbarui', ['class' => 'btn btn-outline-primary']) }}
139 </div>
140 {{ Form::close() }}
141 </div>
142 </div>
143 </div>
144
```

Gambar 4.3. 63 kode program tampilan modal ubah Film

Gambar 4.3.63 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk mengubah data film. Terdapat kode program untuk membuat modal dengan judul “Edit Film” di mana isi dari modal tersebut terdapat *input text*, *number*, *select* pertama, *textarea*, *date*, *select* kedua, tombol “Batal”, dan tombol “Perbarui”. Untuk *input text* akan memunculkan Judul Film tersebut dan dapat diubah atau dibiarkan saja. Untuk *input number* akan memunculkan Durasi tersebut dan dapat diubah atau dibiarkan saja. Untuk *select* pertama akan langsung berisi Rating dari film tersebut. Untuk *input textarea* akan memunculkan Sinopsis tersebut dan dapat diubah atau dibiarkan saja. Untuk *input date* akan memunculkan Jadwal Rilis tersebut dan dapat diubah atau dibiarkan saja. Untuk *select* kedua akan langsung berisi Bahasa dari film tersebut. Tombol “Batal” untuk membatalkan pengubahan film. Tombol “Perbarui” untuk memperbarui atau mengubah data film.

```

150 <script>
151 $('.btn-edit').click(function() {
152 const f = $(this).data('f');
153 const fd = $(this).data('fd');
154 $('#id').val(f.id);
155 $('#film').val(f.film)
156 $('#durasi').val(f.durasi)
157 $('#rating').val(f.rating)
158 $('#sinopsis').val(f.sinopsis)
159 $('#jadwal_rilis').val(fd.jadwal_rilis)
160 $('#bahasa').val(fd.bahasa)
161 })

```

Gambar 4.3. 64 kode program *script edit Film*

Gambar 4.3.64 adalah kode program pada bagian script agar data Judul Film, Durasi, Rating, Sinopsis, Jadwal Rilis, dan Bahasa dari data film yang ingin diubah dapat ditampilkan pada bagian modal “Edit Film”.

```

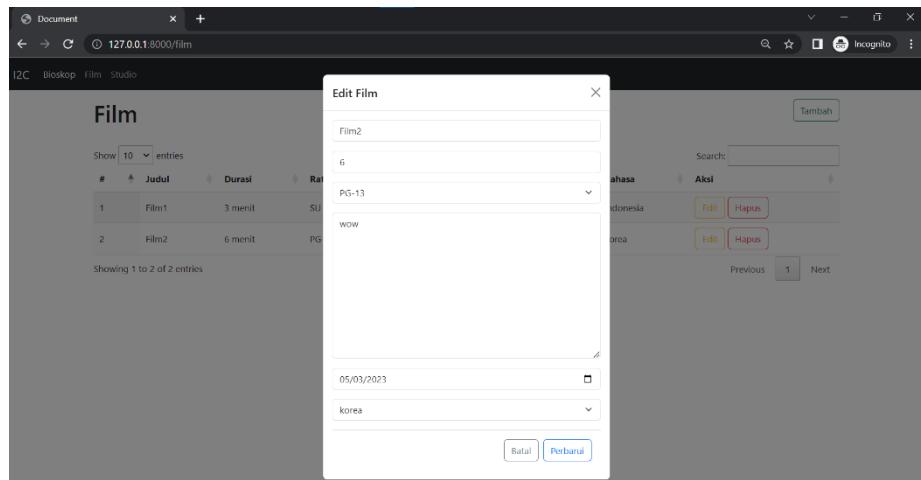
60 public function update_film(Request $request){
61 $film = Film::find($request->id);
62 if($film->film == $request->edit_judul_film){
63 $validation = $request->validate([
64 "edit_judul_film"=>"required"
65]);
66 }else{
67 $validation = $request->validate([
68 "edit_judul_film"=>"required | unique:films,film"
69]);
70 }
71 $filmdetail = FilmDetail::find($request->id);
72 $datafilm = $validation["edit_judul_film"];
73 $datadurasi = $request->edit_durasi;
74 $datarating = $request->edit_rating;
75 $datasinopsis = $request->edit_sinopsis;
76 $datajadwalrilis = $request->edit_jadwal_rilis;
77 $databahasa = $request->edit_bahasa;
78 $film->film=$datafilm;
79 $film->durasi=$datadurasi;
80 $film->rating=$datarating;
81 $film->sinopsis=$datasinopsis;
82 try {
83 $film->update();
84 } catch (QueryException $e) {
85
86 return back()->withErrors(["status"=>"Fail to input data"]);
87 }
88 $filmdetail->film_id=$film->id;
89 $filmdetail->jadwal_rilis=$datajadwalrilis;
90 $filmdetail->bahasa=$databahasa;
91 $filmdetail->update();
92 try {
93 $filmdetail->update();
94 } catch (QueryException $e) {
95
96 return back()->withErrors(["status"=>"Fail to input data"]);
97 }
98 return redirect("/film");

```

Gambar 4.3. 65 kode program *controller ubah Film*

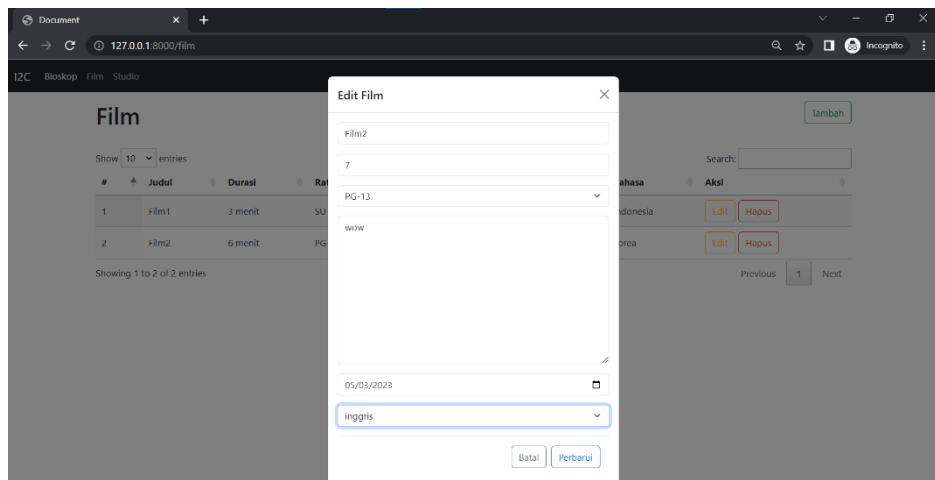
Gambar 4.3.65 adalah kode program yang penulis kerjakan pada *controller* untuk mengubah data film. Nama fungsinya adalah “*update\_film*”. Pada fungsi ini, pertama-tama sistem akan mencari / *find* id yang sesuai dengan data film tersebut. Jika id sudah ditemukan, selanjutnya data-data yang memiliki id film tersebut akan

diubah sesuai dengan yang di-input pada modal. Untuk Judul Film dibuat *unique* agar tidak terjadi duplikat.



Gambar 4.3. 66 tampilan modal ubah Film

Gambar 4.3.66 adalah tampilan ketika tombol “Ubah” pada data film baru diklik. Ketika diklik, akan muncul modal untuk mengubah data film. Seluruh field telah terisi sesuai dengan data dari film baru tersebut.



Gambar 4.3. 67 tampilan mengubah Film

Gambar 4.3.67 adalah tampilan ketika penulis ingin mengubah Durasi menjadi “7” dan Bahasa menjadi “inggris”. Kemudian klik tombol “Perbarui” untuk mengubah data film tersebut.

| # | Judul | Durasi  | Rating | Sinopsis    | Rilis      | Bahasa    | Aksi                                         |
|---|-------|---------|--------|-------------|------------|-----------|----------------------------------------------|
| 1 | Film1 | 3 menit | SU     | Film1 keren | 2023-03-02 | Indonesia | <button>Edit</button> <button>Hapus</button> |
| 2 | Film2 | 7 menit | PG-13  | wow         | 2023-05-03 | inggris   | <button>Edit</button> <button>Hapus</button> |

Gambar 4.3. 68 tampilan Film berhasil diubah

Gambar 4.3.68 adalah tampilan ketika tombol “Perbarui” pada modal diklik dan data film berhasil diubah. Terlihat bahwa data film yang baru diubah tersebut langsung ditampilkan pada tabel.

#### 4.3.3.4 Delete

##### Delete Film

- Munculkan SWAL konfirmasi dgn rincian sbb:
  - Tulisan: "Yakin untuk menghapus Film ini?"
  - Tombol "Batal". Close SWAL.
  - Tombol "Hapus". Ketika di-klik, maka DELETE data di database.
- DELETE data di database:
  - Route: /film
  - Controller: FilmController@delete\_film
  - Method: DELETE
  - Rincian di Controller:
    - Step 1
      - FIND data pada tabel films yg memiliki id = **data-id** dari item yg dipilih.
      - Jika ADA data, maka lanjut ke **Step 2**.
    - Step 2
      - Delete data films ybs di database.
      - Delete data films\_details ybs di database.
    - Step 3
      - Return redirect ke halaman /film.

Gambar 4.3. 69 task delete Theater

Gambar 4.3.69 adalah *task* untuk membuat bagian *delete* pada modul Theater.

```
23 Route::DELETE('/theater', [TheaterController::class, "delete_theater"]);
```

Gambar 4.3. 70 kode program route delete Film

Gambar 4.3.70 adalah kode program yang telah penulis kerjakan untuk membuat *route delete* untuk menghapus data modul Film. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *delete*, karena berfungsi untuk menghapus data. Untuk menghapus data, maka fungsi yang akan dipanggil dalam *controller* bernama “*delete\_film*”.

```

145 <div>
146 {{ Form::open(['method' => 'DELETE', 'id' => 'deleteform']) }}
147 {{ Form::hidden('id', null, ['id' => 'deleteid']) }}
148 {{ Form::close() }}
149 </div>
150 <script>
151 $('.btn-delete').click(function() {
152 Swal.fire({
153 icon: "warning",
154 text: "Yakin untuk menghapus Film ini?",
155 showCancelButton: true,
156 confirmButtonColor: '#3085d6',
157 cancelButtonColor: '#d33',
158 cancelButtonText: 'Batal',
159 confirmButtonText: 'Hapus'
160 }).then((result) => {
161 if (result.isConfirmed) {
162 const id = $(this).data("id");
163 $('#deleteid').val(id);
164 $('#deleteform').submit();
165 }
166 })
167 })

```

Gambar 4.3. 71 kode program tampilan hapus Film

Gambar 4.3.71 kode program yang penulis kerjakan pada *view* untuk membuat tampilan ketika ingin menghapus data film yang dipilih. Untuk tampilannya menggunakan fitur dari SWAL (*Sweet Alert*). Terdapat kode program untuk membuat form *delete*. Kemudian terdapat kode program untuk penggunaan SWAL. Pada SWAL ini terdapat pesan “Yakin untuk menghapus Film ini?”, tombol “Hapus” untuk menghapus film tersebut, serta tombol “Kembali” untuk kembali ke halaman *main page*.

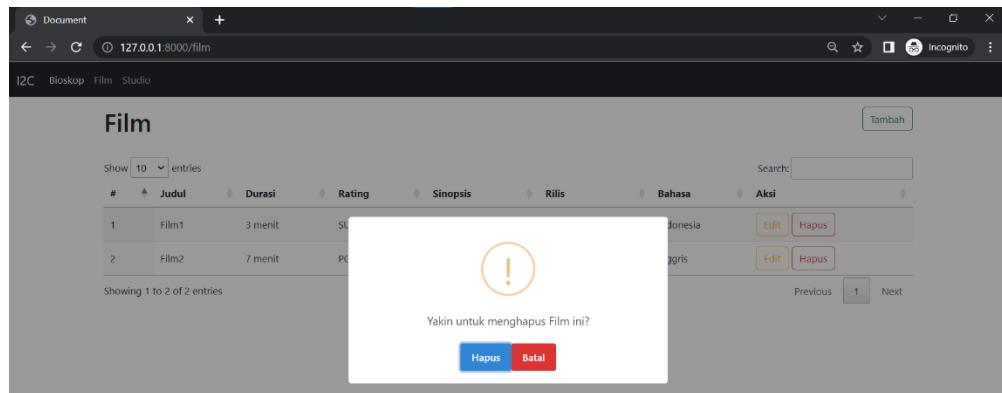
```

100 public function delete_film(Request $request){
101 $id = $request->id;
102 $film = Film::find($id);
103 $filmdetail = FilmDetail::find($id);
104
105 if($film != null && $filmdetail != null){
106 $result = $filmdetail->forceDelete();
107 $result = $film->forceDelete();
108 } else {
109 return back()->withErrors("Data tidak ada");
110 }
111 if($result){
112 return redirect("/film");
113 }else{
114 return back()->withErrors("Fail to Delete");
115 }
116 }

```

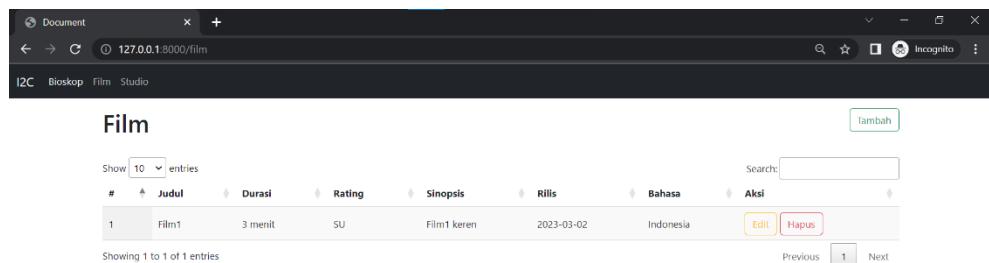
Gambar 4.3. 72 kode program controller hapus Film

Gambar 4.3.72 adalah kode program yang penulis kerjakan pada *controller* untuk menghapus data film. Nama fungsinya adalah “*delete\_film*”. Pada fungsi ini, pertama-tama sistem akan mencari/*find* id yang sesuai dengan data film tersebut. Jika id sudah ditemukan, selanjutnya data pada tabel *films\_details* terlebih dahulu dihapus, karena terdapat foreign key ke tabel *films*. Selanjutnya data pada tabel *films* dihapus.



Gambar 4.3. 73 tampilan SWAL hapus Film

Gambar 4.3.73 adalah tampilan ketika tombol “Hapus” dari data film yang baru diubah diklik. Ketika diklik, maka akan muncul tampilan SWAL yang memunculkan pesan konfirmasi terlebih dahulu sebelum data film tersebut dihapus. Untuk menghapus data tersebut, selanjutnya klik tombol “Hapus”.



Gambar 4.3. 74 tampilan Film berhasil dihapus

Gambar 4.3.74 adalah tampilan ketika tombol “Hapus” pada SWAL diklik. Ketika diklik, data film tersebut dihapus dan tidak muncul kembali pada tabel *films*.

### 4.3.5 CRUD Studio

Untuk modul Studio pertama-tama penulis membuat untuk bagian *read* dahulu, kemudian *create*, *update*, dan terakhir *delete*.

#### 4.3.5.1 Read

**Modul Studio**

**Main Page**

- Route: /studio
- Controller: StudioController@index\_studio
- Method: GET
- View: views/studio.blade.php.
- Template:
- Tampilan dibuat seperti pada gambar berikut:

● Tulisan judul: "Studio".  
 ● Tombol di sebelah kanan atas: Tombol "Tambah". Ketika di-klik, munculkan Modal "Tambah Studio". Lihat bagian "Insert Studio".  
 ● Bagian Filter:  
 ■ Tampilkan select "Theater", Bungkus di dalam sebuah form dengan method GET.  
 ■ Name = "theater".  
 ■ GET dari tabel theaters, order by nama\_bioskop ASCENDING.  
 ■ Value = theaters-id.  
 ■ Table = "[theaters->nama\_bioskop] - [theaters->kota]".  
 ■ Default Selected:  
 ■ Jika pada URL ada parameter "theater", dan TIDAK kosong, maka set item yg memiliki value = value parameter "theater" sebagai default selected.  
 ■ Contoh URL yg dilakukan: "/studio?theater=2".  
 ■ jika pada URL TIDAK ada parameter "theater", atau ada parameter "theater" tetapi KOSONG (Contoh: "/studio/theater="), maka TIDAK ada item yg diberi sebagai default selected.  
 ■ Placeholder: "Filter Theater".  
 ■ Terapkan select2. Lihat rincian penggunaannya di sini: <http://select2.org/get-started/basic-usage>.  
 ■ Karena ada data yg dipilih (on change), maka submit form GET (yg membungkus filter) dg menambahkan jQuery.  
 ● Rincian Tabel:  
 ■ GET data dari tabel studios yg memiliki:  
 ■ jika ADA filter "theater" yg diterapkan, maka filter berdasarkan theater\_id = value selez "theater" yg dipilih.  
 ■ jika TIDAK ada filter "theater" yg diterapkan, maka tidak diterapkan filter apapun, alias tampilkan seluruh data  
 ■ Order by theater\_id ASCENDING.

| # | Theater | Studio | Jenis Studio | Aksi |
|---|---------|--------|--------------|------|
| 1 |         |        |              |      |
| 2 |         |        |              |      |

● Kolom "#":> diisi oleh urutan mulai dari 1 sd/jumlah data.  
 ● Kolom "Theater":> "[theaters->nama\_bioskop] - [theaters->kota]".  
 ● Kolom "Studio":> studios->studio.  
 ● Kolom "Jenis Studio":> studios->jenis\_studio.  
 ● Kolom "Aksi":  
 ■ Tombol "Edit".  
 ■ Sertakan atribut data-e = (studios dari data ybs).  
 ■ Ketika tombol di-klik, munculkan Modal "Edit Studio". Lihat bagian "Update Studio".  
 ■ Tombol "Hapus".  
 ■ Sertakan atribut data-id = (studios->id dari data ybs).  
 ■ Lihat bagian "Delete Studio".

Gambar 4.3. 75 task main page/read Studio

Gambar 4.3.75 adalah *task* untuk membuat *main page* untuk modul Studio.

```
30 Route::GET('/studio', [StudioController::class, "index_studio"]);
```

Gambar 4.3. 76 kode program route read Studio

Gambar 4.3.76 adalah kode program yang telah penulis kerjakan untuk membuat *route read* untuk menampilkan data modul Studio. *Route* ini nantinya

akan menjadi alamat dari halaman web. Metode yang digunakan adalah *get*, karena berfungsi untuk menampilkan data. Untuk menampilkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “index\_studio”.

```

1 @extends('master')
2 @section('content')
3
4

5

6

7 @if ($errors->any())
8 | <div class="alert alert-danger">
9 | | {{ $errors->first() }}
10 | </div>
11 @endif
12 <div class="d-flex justify-content-between">
13 <h1>Studio</h1>
14 <div>
15 | <button data-bs-target="#tambahstudio" data-bs-toggle="modal" class="btn btn-outline-success">Tambah</button>
16 </div>
17

18 {{ Form::open(['method' => 'GET', 'id'=>"filterform"]) }}
19 <div class="mb-3 d-flex gap-3">
20 | {{ Form::select('theater', $theatercoll,$selected, ['class' => 'form-select','placeholder'=>"Filter Theater","id"=>"filter"]) }}
21 </div>
22 {{ Form::close() }}

23 <table id=table_id class="display">
24 <thead>
25 | <th>#</th>
26 | <th>Theater</th>
27 | <th>Studio</th>
28 | <th>Jenis Studio</th>
29 | <th>Aksi</th>
30 </thead>
31 <tbody>
32 | @foreach ($studios as $item)
33 | <tr>
34 | | <td>
35 | | | {{ $loop->iteration }}
36 | | </td>
37 | | <td>
38 | | | {{ $item->studio_theater->nama bioskop}} - {{($item->studio_theater->kota)}}
39 | | </td>
40 | | <td>
41 | | | {{ $item->studio }}
42 | | </td>
43 | | <td>
44 | | | {{ $item->jenis_studio }}
45 | | </td>
46 | | <td>
47 | | | <button class="btn btn-outline-warning btn-edit" data-s="{{ $item }}"
48 | | | | data-bs-target="#updatestudio" data-bs-toggle="modal">Ubah</button>
49 | | | <button class="btn btn-outline-danger btn-delete" data-id="{{ $item->id }}>Hapus</button>
50 | | </td>
51 | </tr>
52 | @endforeach
53 </tbody>
54 </table>
55

```

Gambar 4.3. 77 kode program tampilan Studio

Gambar 4.3.77 adalah kode program yang penulis kerjakan untuk membuat tampilan dari modul Studio. Pada tampilan ini akan menampilkan tabel yang menggunakan DataTables. Tabel ini akan menampilkan data-data dari tabel studios. Terdapat juga kode program *select* untuk *filter theater*. Terdapat juga tombol Ubah untuk mengubah data studio tersebut dan Hapus untuk menghapus data studio tersebut.

```

12 public function index_studio(Request $request)
13 {
14 $filter = $request->theater;
15 if(isset($filter)){
16 $studios = Studio::where('theater_id',$filter)->orderBy('theater_id', 'ASC')->get();
17 }else{
18 $studios = Studio::orderBy('theater_id','ASC')->get();
19 }
20
21 $theaters = Theater::orderBy("nama_bioskop","ASC")->get();
22 $theatercoll = [];
23 foreach($theaters as $item){
24 $theatercoll[$item->id]=$item->nama_bioskop ."-". $item->kota;
25 }
26 $jeniscoll = [];
27 foreach(Studio::JENIS as $item){
28 $jeniscoll[$item] = $item;
29 }
30 return view("studio",["studios"=>$studios,"theatercoll"=>$theatercoll,"jeniscoll"=>$jeniscoll,"selected"=>$filter]);
31 }

```

Gambar 4.3. 78 kode program *controller read Studio*

Gambar 4.3.78 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data dari tabel studios. Data tersebut diurutkan berdasarkan abjad dari kolom theater\_id secara *ascending* atau menurun. Terdapat juga kode program untuk melakukan *filter* theater.

Tabel studios saat ini masih kosong atau belum ada data, sehingga penulis perlu mengisi data *dummy* ke dalam tabel tersebut. Untuk mengisi data *dummy* ke dalam tabel studios, penulis menggunakan salah satu fitur dari Laravel yaitu *seeder*

```

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class StudioSeeder extends Seeder
10 {
11 /**
12 * Run the database seeds.
13 */
14 public function run(): void
15 {
16 DB::table('studios')->insert([
17 "theater_id"=>"2",
18 "jenis_studio"=>"3D",
19 "studio"=>2,
20]);
21 DB::table('studios')->insert([
22 "theater_id"=>"3",
23 "jenis_studio"=>"4X",
24 "studio"=>6,
25]);
26 }
27 }
28

```

Gambar 4.3. 79 kode program *seeder Studio*

Gambar 4.3.79 adalah kode program yang telah penulis kerjakan pada bagian *seeder*. Kolom pada tabel studios yang perlu diisi data *dummy* adalah kolom theater\_id, kolom jenis\_studio, dan kolom studio.

| # | Theater           | Studio | Jenis Studio | Aksi          |
|---|-------------------|--------|--------------|---------------|
| 1 | XXI TSM - Bandung | 2      | 3D           | [Ubah, Hapus] |
| 2 | Asik - Bandung    | 6      | 4X           | [Ubah, Hapus] |

Gambar 4.3. 80 tampilan modul Studio

Gambar 4.3.80 adalah tampilan dari modul Studio. Tampilan ini memiliki judul “Studio”. Lalu terdapat tabel yang berisi kolom untuk penomoran, Theater, Studio, Jenis Studio, serta Aksi di mana kolom ini berisi tombol Ubah dan Hapus. Lalu terdapat select yang berisi daftar theater. Ketika salah satu theater diklik, maka tabel hanya akan menampilkan theater tersebut saja. Selain itu terdapat juga tombol Tambah menampilkan modal untuk menambah data Studio.

| # | Theater           | Studio | Jenis Studio | Aksi          |
|---|-------------------|--------|--------------|---------------|
| 1 | XXI TSM - Bandung | 2      | 3D           | [Ubah, Hapus] |

Gambar 4.3. 81 tampilan modul studio ketika di-filter

Gambar 4.3.81 adalah tampilan dari modul Studio ketika di-filter berdasarkan Theater. Pada gambar tersebut, isi dari *select* adalah “XXI TSM-Bandung” sehingga pada tabel, data yang ditunjukkan hanyalah data studio untuk theater XXI TSM-Bandung.

#### 4.3.5.2 Create

##### Insert Studio

- Munculkan Modal "Tambah Studio" yg berisikan form dgn fields sbb:
  - **Catatan: Terapkan LaravelCollective pada penggunaan Form.**
  - Select "Bioskop".
    - Tampilan data yg sama seperti yg diterapkan pada select "Theater".
    - Value = theaters->id.
    - Teks = "(theaters->nama\_bioskop) - (theaters->kota)".
    - Placeholder: "Pilih Bioskop".
    - Required.
  - Select "Jenis Studio".
    - Data / item disisi oleh const JENIS dari Model Studio.
    - Value = Teks = item dari const JENIS.
    - Placeholder: "Pilih Jenis Studio".
    - Required.
  - Input number "Studio". Required.
  - Modal Footer:
    - Tombol "Batal". Close Modal.
    - Tombol "Simpan". INSERT data ke database.
- INSERT data ke database:
  - Route: /studio
  - Controller: StudioController@store\_studio
  - Method: POST
  - Rincian di Controller:
    - **Step 1**
      - Insert data ke tabel studios:
        - theater\_id = value select "Bioskop" yg dipilih.
        - jenis\_studio = value select "Jenis Studio".
        - studio = value input "Studio".
    - **Step 2**
      - Update data di tabel theaters yg memiliki id = value select "Bioskop" yg dipilih:
        - jumlah\_studio = jumlah data di tabel studios yg memiliki theater\_id = value select "Bioskop" yg dipilih.
    - **Step 3**
      - Return redirect ke halaman /studio.

**Gambar 4.3. 82 task create Studio**

Gambar 4.3.82 adalah *task* untuk membuat bagian *create* atau *insert* pada modul Studio.

```
31 Route::POST('/studio', [StudioController::class, "store_studio"]);
```

**Gambar 4.3. 83 kode program route create Studio**

Gambar 4.3.83 adalah kode program yang telah penulis kerjakan untuk membuat *route create* untuk menambahkan data modul Studio. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *post*, karena berfungsi untuk menambahkan data. Untuk menambahkan data, maka fungsi yang akan dipanggil dalam *controller* bernama “*store\_studio*”.

```

56 <div class="modal fade" tabindex="-1" id="tambahstudio">
57 <div class="modal-dialog">
58 <div class="modal-content">
59 <div class="modal-header">
60 <h5 class="modal-title">Tambah Studio</h5>
61 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
62 </div>
63 <div class="modal-body">
64 {{ Form::open(['method' => 'POST', 'url' => '/studio']) }}
65 {{ Form::select('Bioskop', $theatercoll, null, ['class' => 'form-select', 'placeholder'=>"Pilih Bioskop", "required"]) }}
66 </div>
67 <div class="mb-3">
68 {{ Form::select('Jenis_Studio', $jeniscoll, null, ['class' => 'form-select', 'placeholder' => 'Pilih Jenis Studio', 'required']) }}
69 </div>
70 <div class="mb-3">
71 {{ Form::number('Studio', null, ['class' => 'form-control', 'placeholder' => 'Studio', 'required']) }}
72 </div>
73 <div class="modal-footer">
74 {{ Form::reset('Batal', ['class' => 'btn btn-outline-secondary', 'data-bs-dismiss' => 'modal']) }}
75 {{ Form::submit('Simpan', ['class' => 'btn btn-outline-primary']) }}
76 </div>
77 {{ Form::close() }}
78 </div>
79 </div>
80 </div>
81
82

```

Gambar 4.3. 84 kode program tampilan modal tambah Studio

Gambar 4.3.84 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk menambah studio. Terdapat kode program untuk membuat modal dengan judul “Tambah Studio” di mana isi dari modal tersebut terdapat *select* pertama, *select* kedua, *input number*, tombol “Batal”, dan tombol “Simpan”. Untuk *select* pertama menggunakan fitur dari Select2. Isi dari *select* ini adalah data-data Nama Bioskop dari tabel theaters. Untuk menghindari *input* data kosong, maka *select* ini dibuat menjadi *required*. Untuk *select* kedua menggunakan fitur dari Select2. Isi dari *select* ini adalah data-data dari const JENIS yang terdapat pada model Studio. Untuk menghindari *input* data kosong, maka *select* ini dibuat menjadi *required*. Lalu *input number* ini berfungsi untuk menerima input Studio. Untuk menghindari *input* data kosong, Tombol “Batal” untuk membatalkan penambahan studio. Tombol “Simpan” untuk menambah/menyimpan data studio.

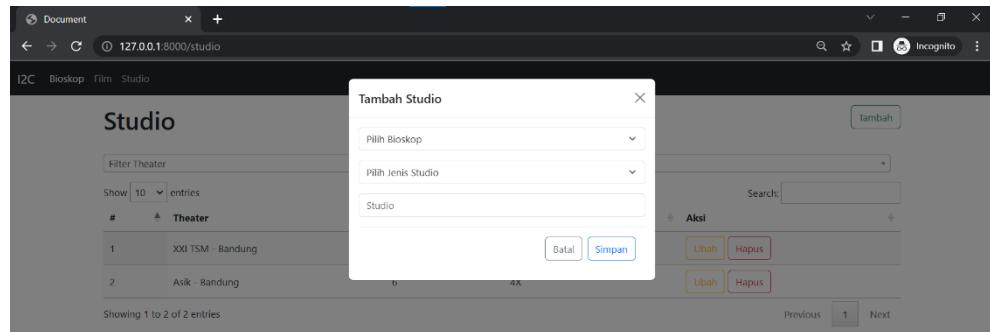
```

32 public function store_studio(Request $request){
33 $datatheaterid = $request->Bioskop;
34 $datajenisstudio = $request->Jenis_Studio;
35 $datastudio = $request->Studio;
36 $studios = new Studio();
37 $studios->theater_id=$datatheaterid;
38 $studios->jenis_studio=$datajenisstudio;
39 $studios->studio=$datastudio;
40 try {
41 $studios->save();
42 } catch (QueryException $e) {
43 return back()->withErrors("Fail to input data");
44 }
45 $theaters = Theater::find($datatheaterid);
46 $theaters->jumlah_studio = count(Studio::where("theater_id",$datatheaterid)->get());
47 try {
48 $theaters->update();
49 } catch (QueryException $e) {
50 return back()->withErrors("Fail to input data");
51 }
52 }
53 return redirect("/studio");
54 }

```

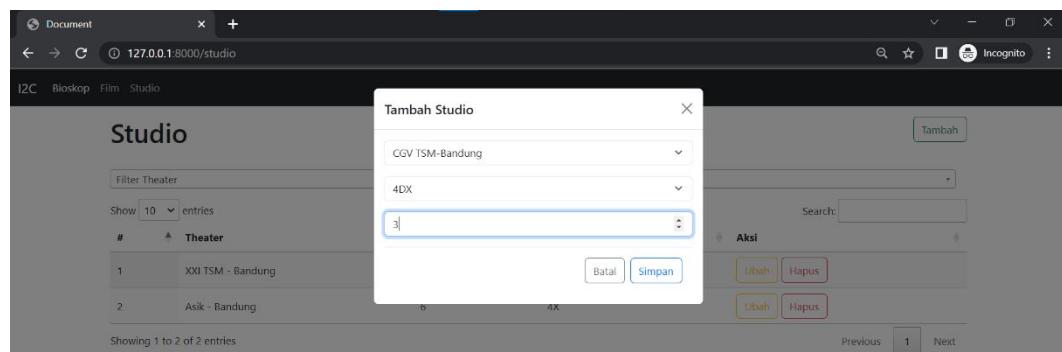
Gambar 4.3. 85 kode program controller tambah Studio

Gambar 4.3.85 adalah kode program yang penulis kerjakan pada *controller* untuk menambahkan data studio. Nama fungsinya adalah “store\_studio”. Pada fungsi ini, data-data yang diinput dari modal akan ditambahkan ke dalam tabel studios. Lalu untuk theater yang di-*input*, Jumlah Studio pada modul Theater akan berubah.



Gambar 4.3. 86 tampilan modal tambah Studio

Gambar 4.3.86 adalah tampilan modal untuk menambahkan data studio. Tampilan modal ini akan muncul setelah tombol “Tambah” diklik. Modal tersebut menggunakan fitur dari Bootstrap.



Gambar 4.3. 87 tampilan menambah Studio

Gambar 4.3.87 adalah tampilan ketika penulis ingin menambahkan data baru, yaitu “CGV TSM-Bandung” sebagai Nama Bioskop, “4DX” sebagai Jenis Studio, dan “3” sebagai Studio. Kemudian klik “Simpan” untuk menyimpan atau menambah data studio baru tersebut.

| # | Theater           | Studio | Jenis Studio | Aksi                                         |
|---|-------------------|--------|--------------|----------------------------------------------|
| 1 | CGV TSM - Bandung | 3      | 4DX          | <button>Ubah</button> <button>Hapus</button> |
| 2 | XXI TSM - Bandung | 2      | 3D           | <button>Ubah</button> <button>Hapus</button> |
| 3 | Asik - Bandung    | 6      | 4K           | <button>Ubah</button> <button>Hapus</button> |

**Gambar 4.3. 88 tampilan Studio berhasil ditambah**

Gambar 4.3.88 adalah tampilan ketika tombol “Simpan” pada modal diklik dan data studio baru berhasil ditambahkan ke dalam tabel studio. Terlihat bahwa data studio baru tersebut langsung ditampilkan pada tabel.

| # | Bioskop | Kota    | Jumlah Studio | Aksi                                         |
|---|---------|---------|---------------|----------------------------------------------|
| 1 | Asik    | Bandung | 0             | <button>Ubah</button> <button>Hapus</button> |
| 2 | CGV TSM | Bandung | 1             | <button>Ubah</button> <button>Hapus</button> |
| 3 | XXI TSM | Bandung | 0             | <button>Ubah</button> <button>Hapus</button> |

**Gambar 4.3. 89 tampilan jumlah studio bertambah**

Gambar 4.3.89 adalah tampilan Jumlah Studio pada modul Theater juga bertambah untuk *theater* CGV TSM – Bandung.

### 4.3.5.3 Update

**Update Studio**

- \* Munculkan Modal "Edit Studio" yg berisikan form dgn fields sbb:
  - o **Catatanc Terapkan LaravelCollective pada penggunaan Form.**
  - o Input hidden dgn value = **data-s->id** dari data ybs.
  - o Select "Edit Bioskop".
    - Tampilkan data yg sama seperti yg diterapkan pada select "Theater".
    - Value = **theaters->id**.
    - Teks = "**(theaters->nama\_bioskop**) - (**theaters->kota**)".
    - Placeholder: "Pilih Bioskop".
    - Default selected = **data-s->theater\_id** dari data ybs.
    - Required.
  - o Select "Edit Jenis Studio".
    - Data / item diisi oleh const JENIS dari Model Studio.
    - Value = Teks = item dari const JENIS.
    - Placeholder: "Pilih Jenis Studio".
    - Default selected = **data-s->jenis\_studio** dari data ybs.
    - Required.
  - o Input number "Edit Studio".
    - Default Value = **data-s->studio**.
    - Required.
  - o Modal Footer:
    - Tombol "Batal". Close Modal.
    - Tombol "Perbarui". UPDATE data di database.
- \* **UPDATE** data di database:
  - o Route: /studio
  - o Controller: StudioController@update\_studio
  - o Method: PATCH
  - o Rincian di Controller:
    - **Step 1**
      - FIND data pada tabel studios yg memiliki id = value input hidden.
    - **Step 2**
      - Update data di database:
        - **theater\_id** = value select "Edit Bioskop".
        - **jenis\_studio** = value select "Edit Jenis Studio".
        - **studio** = value input "Studio".
    - **Step 3**
      - Jika studios->theater\_id (ketika belum di-update) BERBEDA dengan value select "Edit Bioskop", maka:
        - Update data di tabel theaters yg memiliki id = studios->theater\_id (ketika belum di-update).
          - jumlah\_studio = jumlah data di tabel studios yg memiliki theater\_id = studios->theater\_id (ketika belum di-update).
        - Update data di tabel theaters yg memiliki id = value select "Edit Bioskop".
          - jumlah\_studio = jumlah data di tabel studios yg memiliki theater\_id = value select "Edit Bioskop".
    - **Step 4**
      - Return redirect ke halaman /studio.

- Jika studios->theater\_id (ketika belum di-update) BERBEDA dengan value select "Edit Bioskop", maka:
  - Update data di tabel theaters yg memiliki id = studios->theater\_id (ketika belum di-update).
    - jumlah\_studio = jumlah data di tabel studios yg memiliki theater\_id = studios->theater\_id (ketika belum di-update).
  - Update data di tabel theaters yg memiliki id = value select "Edit Bioskop".
    - jumlah\_studio = jumlah data di tabel studios yg memiliki theater\_id = value select "Edit Bioskop".
- **Step 4**
  - Return redirect ke halaman /studio.

**Gambar 4.3. 90 task update Studio**

Gambar 4.3.90 adalah *task* untuk membuat bagian *update* pada modul Studio.

```
32 Route::PATCH('/studio', [StudioController::class, "update_studio"]);
```

**Gambar 4.3. 91 kode program route update Studio**

Gambar 4.3.91 adalah kode program yang telah penulis kerjakan untuk membuat *route update* untuk mengubah data modul Studio. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *patch*, karena

berfungsi untuk mengubah data. Untuk mengubah data, maka fungsi yang akan dipanggil dalam *controller* bernama “update\_studio”.

```

83 <div class="modal fade" tabindex="-1" id="updatestudio">
84 <div class="modal-dialog">
85 <div class="modal-content">
86 <div class="modal-header">
87 <h5 class="modal-title">Edit Studio</h5>
88 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
89 </div>
90 <div class="modal-body">
91 {{ Form::open(['method' => 'PATCH']) }}
92 {{ Form::hidden('id', null, ['id' => 'id']) }}
93 <div class="mb-3">
94 {{ Form::select('Edit_Bioskop', $theatercoll, null, ['class' => 'form-select', 'placeholder' => 'Pilih Bioskop', 'required', 'id'=>"nama_bioskop"]) }}
95 </div>
96 <div class="mb-3">
97 {{ Form::select('Edit_Jenis_Studio', $jeniscoll, null, ['class' => 'form-select', 'placeholder' => 'Pilih Jenis Studio', 'required','id"=>"jenis_studio"])}}
98 </div>
99 <div class="mb-3">
100 {{ Form::number('Edit_Studio', null, ['class' => 'form-control', 'placeholder' => 'Studio', 'required','id"=>"studio"])}}
101 </div>
102 <div class="modal-footer">
103 {{ Form::reset('Batal', ['class' => 'btn btn-outline-secondary', 'data-bs-dismiss' => 'modal']) }}
104 {{ Form::submit('Perbarui', ['class' => 'btn btn-outline-primary']) }}
105 </div>
106 <div>
107 {{ Form::close()}}
108 </div>
109 </div>
110 </div>

```

Gambar 4.3.92 kode program tampilan ubah Studio

Gambar 4.3.92 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk mengubah data studio. Terdapat kode program untuk membuat modal dengan judul “Edit Studio” di mana isi dari modal tersebut terdapat *select* pertama, *select* kedua, *input number*, tombol “Batal”, dan tombol “Perbarui”. Untuk *select* pertama akan memunculkan Nama Bioskop tersebut. Nama Bioskop tersebut dapat diubah atau dibiarkan saja. Untuk *select* kedua akan memunculkan Jenis Studio tersebut. Untuk *input number* akan memunculkan Studio tersebut dan dapat diubah atau dibiarkan saja. Tombol “Batal” untuk membatalkan pengubahan studio. Tombol “Perbarui” untuk memperbarui atau mengubah data studio.

```

120 $('.btn-edit').click(function() {
121 const s = $(this).data('s');
122 $('#id').val(s.id);
123 $('#nama_bioskop').val(s.theater_id);
124 $('#jenis_studio').val(s.jenis_studio)
125 $('#studio').val(s.studio)
126 })

```

Gambar 4.3.93 kode script ubah Studio

Gambar 4.3.93 adalah kode program pada bagian *script* agar data Nama Bioskop, Jenis Studio, dan Studio dapat ditampilkan pada bagian modal “Edit Studio”.

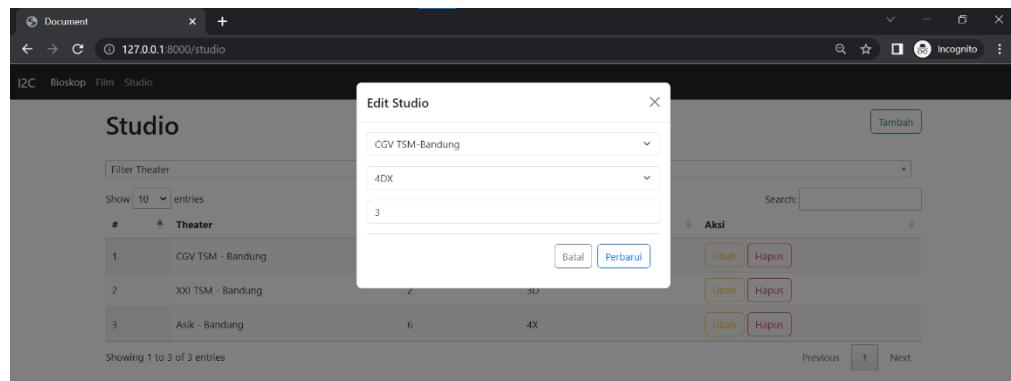
```

55 public function update_studio(Request $request){
56 $studios = Studio::find($request->id);
57 $oldid = $studios->theater_id;
58 $datatheaterid = $request->Edit_Bioskop;
59 $datajenissudio = $request->Edit_Jenis_Studio;
60 $datastudio = $request->Edit_Studio;
61 $studios->theater_id=$datatheaterid;
62 $studios->jenis_studio=$datajenissudio;
63 $studios->studio=$datastudio;
64 try {
65 $studios->update();
66 } catch (QueryException $e) {
67 return back()->withErrors(["status"=>"Fail to update data"]);
68 }
69 if($oldid != $datatheaterid){
70 $theatersold = Theater::find($oldid);
71 $theatersold->jumlah_studio=count(Studio::where("theater_id",$oldid)->get());
72 $theaters = Theater::find($datatheaterid);
73 $theaters->jumlah_studio=count(Studio::where("theater_id",$datatheaterid)->get());
74 try {
75 $theatersold->update();
76 $theaters->update();
77 } catch (QueryException $e) {
78 return back()->withErrors(["status"=>"Fail to update data"]);
79 }
80 }
81 }
82 return redirect("/studio");
83 }

```

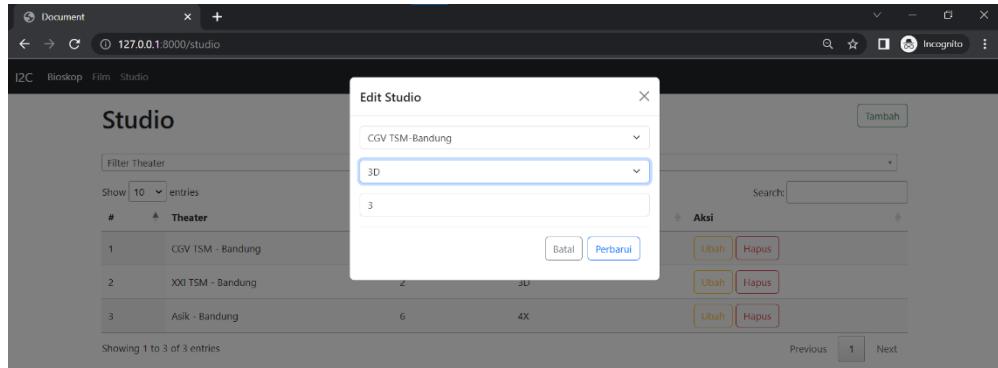
Gambar 4.3. 94 kode program *controller* ubah Studio

Gambar 4.3.94 adalah kode program yang penulis kerjakan pada *controller* untuk mengubah data studio. Nama fungsinya adalah “*update\_studio*”. Pada fungsi ini, pertama-tama sistem akan mencari / *find* id yang sesuai dengan data studio tersebut. Jika id sudah ditemukan, selanjutnya Nama Bioskop Jenis Studio, dan Studio dari id studio tersebut akan diubah sesuai dengan yang di-*input* pada modal.



Gambar 4.3. 95 tampilan modal ubah Studio

Gambar 4.3.95 adalah tampilan ketika tombol “Ubah” pada data studio baru diklik. Ketika diklik, muncul modal untuk mengubah data studio. Pada bagian *input text* dan *select* langsung terisi Nama Bioskop yaitu “CGV TSM-Bandung”, Jenis Studio yaitu “4DX”, Studio yaitu “3”.



Gambar 4.3. 96 tampilan mengubah Studio

Gambar 4.3.96 adalah tampilan ketika penulis ingin mengubah Jenis Studio menjadi “3D”. Kemudian klik tombol “Perbarui” untuk mengubah data studio tersebut.

| # | Theater           | Studio | Jenis Studio | Aksi                                       |
|---|-------------------|--------|--------------|--------------------------------------------|
| 1 | CGV TSM - Bandung | 3      | 3D           | <a href="#">Ubah</a> <a href="#">Hapus</a> |
| 2 | XXI TSM - Bandung | 2      | 3D           | <a href="#">Ubah</a> <a href="#">Hapus</a> |
| 3 | Asik - Bandung    | 6      | 4X           | <a href="#">Ubah</a> <a href="#">Hapus</a> |

Gambar 4.3. 97 tampilan Studio berhasil diubah

Gambar 4.3.97 adalah tampilan ketika tombol “Perbarui” pada modal diklik dan data studio berhasil diubah. Terlihat bahwa data studio yang baru diubah tersebut langsung ditampilkan pada tabel.

#### 4.3.5.4 Delete

##### Delete Studio

- Munculkan SWAL konfirmasi dgn rincian sbb:
  - Tulisan: "Yakin untuk menghapus Studio ini?"
  - Tombol "Batal". Close SWAL.
  - Tombol "Hapus". Ketika di-klik, maka DELETE data di database.
- DELETE data di database:
  - Route: /studio
  - Controller: StudioController@delete\_studio
  - Method: DELETE
  - Rincian di Controller:
    - **Step 1**
      - FIND data pada tabel studios yg memiliki id = **data-id** dari item yg dipilih.
      - Jika ADA data, maka lanjut ke **Step 2**.
    - **Step 2**
      - Delete data studios ybs di database.
    - **Step 3**
      - Update data di tabel theaters yg memiliki id = studios->theater\_id (dari data yg telah dihapus):
        - jumlah\_studio = jumlah data di tabel studios yg memiliki theater\_id = studios\_theater\_id (dari data yg telah dihapus).
    - **Step 4**
      - Return redirect ke halaman /studio.

**Gambar 4.3. 98 task delete Studio**

Gambar 4.3.98 adalah *task* untuk membuat bagian *delete* pada modul Theater.

```
33 Route::DELETE('/studio', [StudioController::class, "delete_studio"]);
```

**Gambar 4.3. 99 kode program route delete Studio**

Gambar 4.3.99 adalah kode program yang telah penulis kerjakan untuk membuat *route delete* untuk menghapus data modul Studio. *Route* ini nantinya akan menjadi alamat dari halaman web. Metode yang digunakan adalah *delete*, karena berfungsi untuk menghapus data. Untuk menghapus data, maka fungsi yang akan dipanggil dalam *controller* bernama “*delete\_studio*”.

```

111 | <div>
112 | {{ Form::open(['method' => 'DELETE', 'id' => 'deleteform']) }}
113 | {{ Form::hidden('id', null, ['id' => 'deleteid']) }}
114 | {{ Form::close() }}
115 | </div>
116 | <script>
117 | $(document).ready(function() {
118 | $('#filter').select2();
119 | });
120 | $('.btn-delete').click(function() {
121 | Swal.fire({
122 | icon: "warning",
123 | text: "Yakin untuk menghapus Studio ini?",
124 | showCancelButton: true,
125 | confirmButtonColor: '#3085d6',
126 | cancelButtonColor: '#d33',
127 | cancelButtonText: 'Batal',
128 | confirmButtonText: 'Hapus'
129 |
130 | }).then((result) => {
131 | if (result.isConfirmed) {
132 | const id = $(this).data("id");
133 | $("#deleteid").val(id);
134 | $('#deleteform').submit();
135 | }
136 | })
137 | })
138 |

```

Gambar 4.3. 100 kode program tampilan hapus Studio

Gambar 4.3.100 kode program yang penulis kerjakan pada *view* untuk membuat tampilan ketika ingin menghapus data studio yang dipilih. Untuk tampilannya menggunakan fitur dari SWAL (*Sweet Alert*). Terdapat kode program untuk membuat form *delete*. Kemudian terdapat kode program untuk penggunaan SWAL. Pada SWAL ini terdapat pesan “Yakin untuk menghapus Studio ini?”, tombol “Hapus” untuk menghapus studio tersebut, serta tombol “Kembali” untuk kembali ke halaman *main page*.

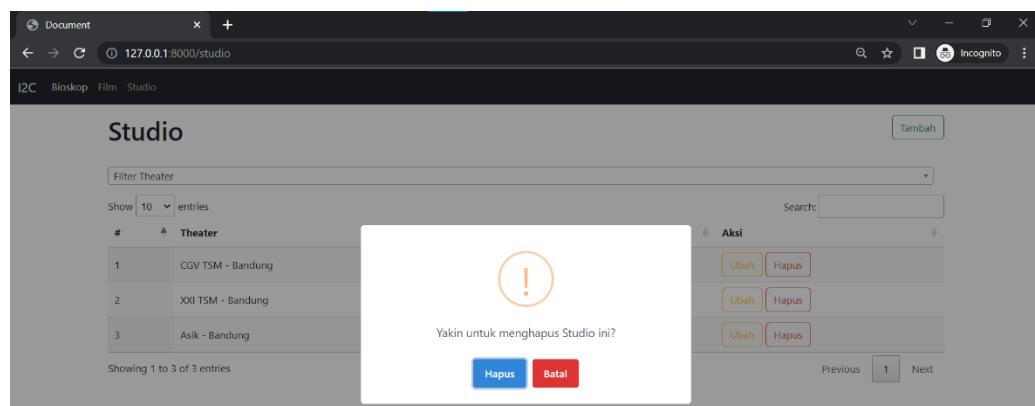
```

84 | public function delete_studio(Request $request){
85 | $id = $request->id;
86 | $studios = Studio::find($id);
87 | $oldid = $studios->theater_id;
88 | if($studios != null){
89 | $result = $studios->delete();
90 | }
91 | $theaters = Theater::find($oldid);
92 | $theaters->jumlah_studio = count(Studio::where("theater_id", $oldid)->get());
93 | $result2 = $theaters->update();
94 | if($result && $result2){
95 | return redirect("/studio");
96 | }else{
97 | return back()->withErrors("Fail to Delete");
98 | }
99 |
100 }

```

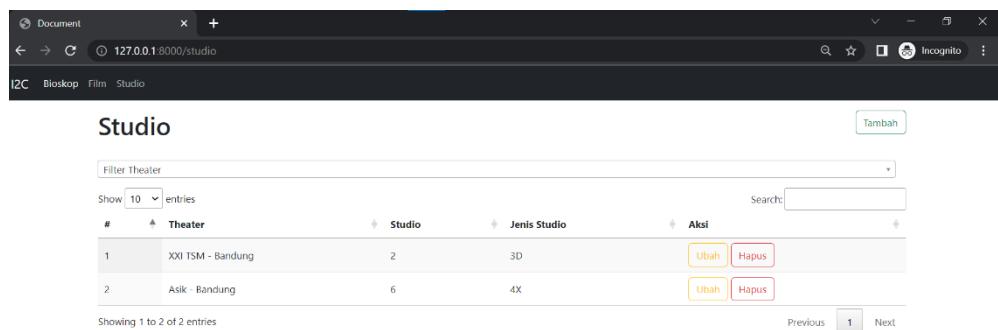
Gambar 4.3. 101 kode program controller hapus Studio

Gambar 4.3.101 adalah kode program yang penulis kerjakan pada *controller* untuk menghapus data studio. Nama fungsinya adalah “`delete_studio`”. Pada fungsi ini, pertama-tama sistem akan mencari/*find* id yang sesuai dengan data studio tersebut. Jika id sudah ditemukan, selanjutnya data theater tersebut akan dihapus dari tabel theaters. Lalu Jumlah Studio dari tabel theaters juga di-*update* kembali karena ketika studio dihapus, maka jumlah pada Jumlah Studio dari tabel theaters akan berubah.



**Gambar 4.3. 102 tampilan SWAL hapus Studio**

Gambar 4.3.102 adalah tampilan ketika tombol “Hapus” dari data theater yang baru diubah diklik. Ketika diklik, maka akan muncul tampilan SWAL yang memunculkan pesan konfirmasi terlebih dahulu sebelum data studio tersebut dihapus. Untuk menghapus data tersebut, selanjutnya klik tombol “Hapus”.



**Gambar 4.3. 103 tampilan Studio berhasil dihapus**

Gambar 4.3.103 adalah tampilan ketika tombol “Hapus” pada SWAL diklik. Ketika diklik, data studio tersebut dihapus dan tidak muncul kembali pada tabel studios.

| # | Bioskop | Kota    | Jumlah Studio | Aksi           |
|---|---------|---------|---------------|----------------|
| 1 | Asik    | Bandung | 0             | [Ubah] [Hapus] |
| 2 | CGV TSM | Bandung | 0             | [Ubah] [Hapus] |
| 3 | XXI TSM | Bandung | 0             | [Ubah] [Hapus] |

Gambar 4.3. 104 tampilan jumlah studio berubah

Gambar 4.3.104 tampilan pada modul Theater di mana Jumlah Studio untuk CGV TSM telah berubah.

#### 4.4 CoreUI

Setelah mengumpulkan tes berupa *mini project*, penulis menunggu untuk diperiksa terlebih dahulu, jika tes tersebut lulus, maka penulis akan diundang ke dalam Trello untuk membantu mengerjakan proyek SIPPM. Selama menunggu, penulis disarankan oleh Pak Julio Narabel dari pihak I2C Studio untuk melihat fitur-fitur yang terdapat pada CoreUI. *Template* CoreUI ini akan dipakai pada proyek SIPPM. Hal pertama yang dilakukan adalah melakukan *clone* dengan Github Desktop dari tautan yang telah diberikan oleh Pak Julio Narabel. Setelah melakukan *clone*, penulis terlebih dahulu membuat database untuk CoreUI ini pada PostgreSQL. Untuk CoreUI ini, penulis membuat database bernama “coreui”. Kemudian file .env.example disalin terlebih dahulu. Hasil salinannya diberi nama .env. Lalu mengubah kode program pada bagian .env.

```

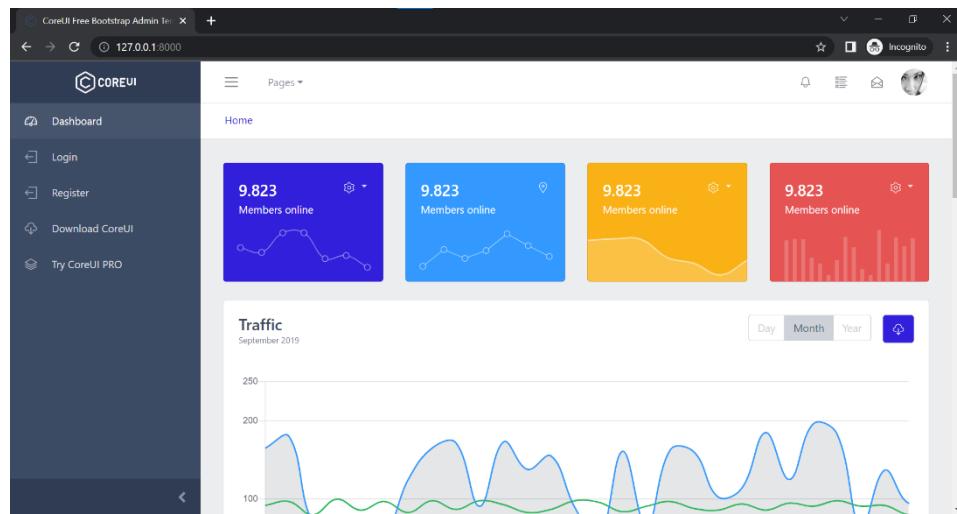
9 DB_CONNECTION=pgsql
10 DB_HOST=127.0.0.1
11 DB_PORT=5432
12 DB_DATABASE=coreui
13 DB_USERNAME=postgres
14 DB_PASSWORD=
15

```

Gambar 4.4. 1 kode program .env  
CoreUI

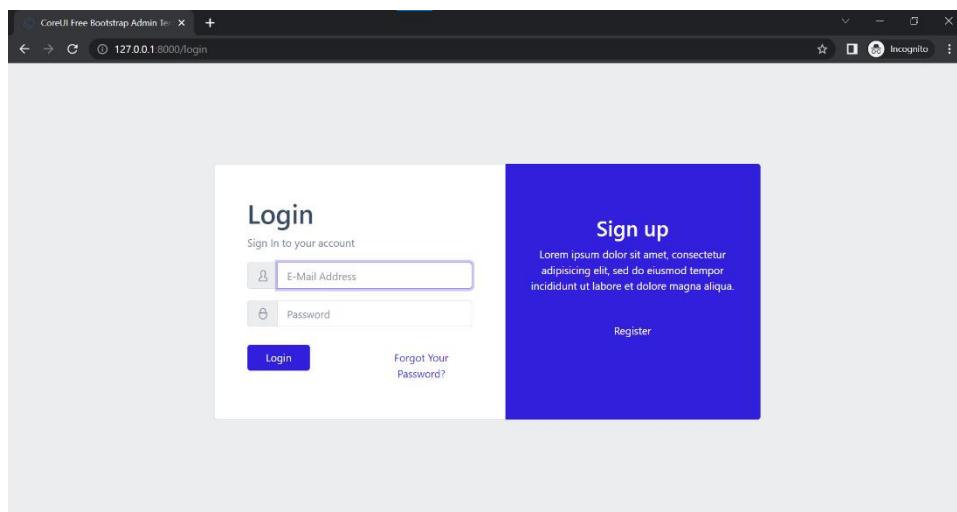
Gambar 4.4.1 adalah kode program yang penulis kerjakan pada file .env agar database pada PostgreSQL dapat terhubung dengan projek *template* CoreUI.

Setelah itu pada file composer.json, versi php disesuaikan dengan yang sudah ter-install. Kemudian pada Command Prompt mengetik perintah “composer update” lalu “npm install” lalu “npm audit fix” lalu “npm run dev” 2 kali. Lalu lakukan seeding dengan mengetik “php artisan:migrate --seed” dan terakhir “php artisan serve”.



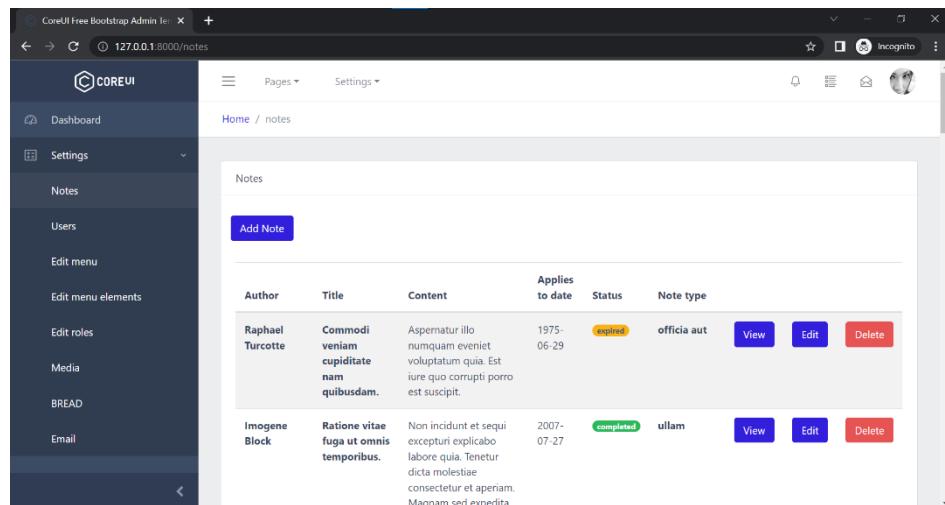
Gambar 4.4. 2 tampilan awal CoreUI

Gambar 4.4.2 adalah tampilan awal dari *template* CoreUI. Pada CoreUI ini terdapat pilihan menu yang dapat dilihat dari *navigation bar* di sebelah kiri. Untuk melihat pilihan menu yang lain, penulis perlu melakukan *login* terlebih dahulu sebagai admin. Caranya adalah klik tab “Login” pada *navigation bar* di sebelah kiri.



Gambar 4.4. 3 tampilan *login* CoreUI

Gambar 4.4.3 adalah tampilan halaman *login* pada *template* CoreUI. Agar dapat *login*, penulis perlu untuk memasukkan *E-mail Address* yaitu “[admin@admin.com](mailto:admin@admin.com)” dan *Password* yaitu “password”. Lalu klik tombol “Login”.



Gambar 4.4. 4 tampilan berhasil *login* CoreUI

Gambar 4.4.4 adalah tampilan yang muncul ketika berhasil *login*. Pada *navigation bar* sebelah kiri terlihat lebih banyak lagi menu yang ditampilkan. Selama menuggu, penulis melihat menu-menu yang terdapat pada CoreUI tersebut.

#### 4.5 Migrasi dan Model Modul Penelitian

Ketika hasil tes sudah diumumkan dan dinyatakan lulus, penulis diundang untuk bergabung dalam Trello untuk penggerjaan proyek SIPP. Pada awal ketika masuk ke dalam penggerjaan proyek, penulis membuat kode program untuk migrasi dan model modul peneltitian. Terdapat beberapa kode program yang telah penulis kerjakan untuk migrasi dan model pada modul penelitian. Kode program tersebut dibuat berdasarkan *task* yang diberikan Bapak Daniel Septian kepada penulis melalui Trello. Pertama-tama hal yang harus dilakukan adalah membuat database baru pada PostgreSQL kemudian menghubungkannya dengan proyek SIPP.

```

9 DB_CONNECTION=pgsql
10 DB_HOST=127.0.0.1
11 DB_PORT=5432
12 DB_DATABASE=lppm
13 DB_USERNAME=postgres
14 DB_PASSWORD=

```

**Gambar 4.5. 1 Kode program .env  
SIPPM**

Gambar 4.5.1 adalah kode program yang penulis kerjakan untuk menghubungkan PostgreSQL dengan proyek SIPPM

#### 4.5.1 Migrasi dan Model Skema Penelitian

##### Tabel mst\_research\_schemes

- Nama Migrasi: `create_mst_research_schemes_table`.
- Nama Tabel: `mst_research_schemes`.
- Rincian Kolom:
  - `id`
  - `research_scheme => string => unique`
  - `status => boolean => default false`
  - `timestamps()`
  - `softDeletes()`

**Gambar 4.5. 2 Task migrasi Skema Penelitian**

Gambar 4.5.1 adalah *task* untuk membuat migrasi Skema Penelitian. Pada *task* ini, penulis membuat migrasi dan untuk tabel `mst_research_schemes`.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMstResearchSchemesTable extends Migration
8 {
9
10 /**
11 * Run the migrations.
12 *
13 * @return void
14 */
15 public function up()
16 {
17 Schema::create('mst_research_schemes', function (Blueprint $table) {
18 $table->id();
19 $table->string("research_scheme")->unique();
20 $table->boolean("status")->default(false);
21 $table->timestamps();
22 $table->softDeletes();
23 });
24 }
25
26 /**
27 * Reverse the migrations.
28 *
29 * @return void
30 */
31 public function down()
32 {
33 Schema::dropIfExists('mst_research_schemes');
34 }
35 }
```

**Gambar 4.5. 3 Kode Program migrasi Skema Penelitian**

Gambar 4.5.3 adalah kode program yang penulis kerjakan untuk migrasi tabel mst\_research\_schemes. Terdapat kode program yang berisi kolom-kolom tabel yang diperlukan sesuai dengan *task*.

- Rincian Model:
  - Model: /Master/ResearchScheme

**Gambar 4.5. 4 Task model Skema Penelitian**

Gambar 4.5.4 adalah *task* untuk membuat model Skema Penelitian bernama ResearchScheme.

- Relasi di Model Master/ResearchScheme:
  - hasOne:
    - Nama Relasi: researchscheme\_rrlecturer.
    - Kolom Terkait: research\_scheme\_id.
    - Model Target: /Master/Requirement/ResearchReqLecturer.

**Gambar 4.5. 5 Task relasi dengan Prasyarat Dosen Penelitian**

Gambar 4.5.5 adalah *task* untuk membuat relasi antara Skema Penelitian dengan Prasyarat Dosen Penelitian.

- o Relasi di Model /Master/ResearchScheme:
  - hasOne:
    - Nama Relasi: researchscheme\_rrdetail.
    - Kolom Terkait: research\_scheme\_id.
    - Model Target: /Master/Requirement/ResearchReqDetail.

**Gambar 4.5. 6 Task relasi dengan Prasyarat Detail Penelitian**

Gambar 4.5.6 adalah *task* untuk membuat relasi antara Skema Penelitian dengan Prasyarat Detail Penelitian.

- Rincian Model:
  - o Relasi di Model Master/ResearchScheme:
    - belongsToMany:
      - Nama Relasi: researchscheme\_studentprogramme.
      - Kolom Terkait: research\_scheme\_id.
      - Model Target: Master/StudentProgramme.

**Gambar 4.5. 7 Task relasi dengan Prasyarat Program Penelitian**

Gambar 4.5.7 adalah *task* untuk membuat relasi antara Skema Penelitian dengan Prasyarat Program Penelitian.

- o Relasi di Model Master/ResearchScheme:
  - hasMany:
    - Nama Relasi: researchscheme\_rroutput.
    - Kolom Terkait: research\_scheme\_id.
    - Model Target: /Master/Requirement/ResearchReqOutput.

**Gambar 4.5. 8 Task relasi dengan Prasyarat Luaran Penelitian**

Gambar 4.5.8 adalah *task* untuk membuat relasi antara Skema Penelitian dengan Prasyarat Luaran Penelitian.

- o Relasi di Model /Master/ResearchScheme:
  - hasOne:
    - Nama Relasi: researchscheme\_rrbudget.
    - Kolom Terkait: research\_scheme\_id.
    - Model Target: /Master/Requirement/ResearchReqBudget.

**Gambar 4.5. 9 Task relasi dengan Prasyarat Anggaran Penelitian**

Gambar 4.5.9 adalah *task* untuk membuat relasi antara Skema Penelitian dengan Prasyarat Luaran Penelitian.

```

1 <?php
2
3 namespace App\Models\Master;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\SoftDeletes;
8 use App\Models\Master\Requirement\ResearchReqLecturer;
9 use App\Models\Master\Requirement\ResearchReqDetail;
10 use App\Models\Master\Requirement\ResearchReqBudget;
11 use App\Models\Master\StudentProgramme;
12
13 class ResearchScheme extends Model
14 {
15 use HasFactory;
16 use SoftDeletes;
17 protected $fillable = [
18 'research_scheme',
19 'status'
20];
21 protected $table = "mst_research_schemes";
22 public function researchscheme_rlecturer(){
23 return $this->hasOne(ResearchReqLecturer::class,"research_scheme_id");
24 }
25 public function researchscheme_rrdetail(){
26 return $this->hasOne(ResearchReqDetail::class,"research_scheme_id");
27 }
28 public function researchscheme_rrbudget(){
29 return $this->hasOne(ResearchReqBudget::class,"research_scheme_id");
30 }
31 public function researchscheme_studentprogramme(){
32 return $this->belongsToMany(StudentProgramme::class,"mst_research_req_programmes","research_scheme_id","student_programme_id");
33 }
34 public function researchscheme_routput(){
35 return $this->hasMany(ResearchReqOutput::class,"research_scheme_id");
36 }
37 }

```

**Gambar 4.5. 10 Kode Program model Skema Penelitian**

Gambar 4.5.10 adalah kode program yang penulis kerjakan untuk model ResearchScheme. Terdapat kode program untuk kolom tabel yang dapat diisi secara manual serta relasi tabel yang diperlukan.

#### 4.5.2 Migrasi dan Model Prasyarat Dosen Penelitian

##### Tabel mst\_research\_req\_lecturers

- Nama Migrasi: create\_mst\_research\_req\_lecturers\_table.
- Nama Tabel: mst\_research\_req\_lecturers.
- Rincian Kolom:
  - research\_scheme\_id => unsigned integer => Primary Key => FK ke tabel mst\_research\_schemes.
  - lecturer\_reqs => string => nullable
  - min\_lecturer => integer => default 0
  - max\_lecturer => integer => nullable
  - min\_lead\_position\_id => unsigned integer => nullable
  - max\_lead\_position\_id => unsigned integer => nullable
  - min\_lead\_position\_strata => string => nullable
  - max\_lead\_position\_strata => string => nullable
  - maximum\_age => integer => nullable
  - min\_member => integer => default 0
  - min\_member\_position\_id => unsigned integer => nullable
  - min\_member\_position\_strata => string => nullable
  - min\_partner => integer => default 0
  - min\_partner\_position\_id => unsigned integer => nullable
  - min\_partner\_position\_strata => string => nullable
  - timestamps()
  - softDeletes()

**Gambar 4.5. 11 Task migrasi Prasyarat Dosen Penelitian**

Gambar 4.5.11 adalah *task* untuk membuat migrasi Prasyarat Dosen Penelitian. Pada *task* ini, penulis membuat migrasi untuk tabel mst\_research\_req\_lecturer.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMstResearchReqLecturersTable extends Migration
8 {
9 /**
10 * Run the migrations.
11 *
12 * @return void
13 */
14 public function up()
15 {
16 Schema::create('mst_research_req_lecturers', function (Blueprint $table) {
17 $table->unsignedInteger('research_scheme_id')->primary();
18 $table->foreign('research_scheme_id')->references("id")->on("mst_research_schemes");
19 $table->string("lecturer_reqs")->nullable();
20 $table->integer("min_lecturer")->default(0);
21 $table->integer("max_lecturer")->nullable();
22 $table->unsignedInteger("min_lead_position_id")->nullable();
23 $table->unsignedInteger("max_lead_position_id")->nullable();
24 $table->string("min_lead_position_strata")->nullable();
25 $table->string("max_lead_position_strata")->nullable();
26 $table->integer("maximum_age")->nullable();
27 $table->integer("min_member")->default(0);
28 $table->unsignedInteger("min_member_position_id")->nullable();
29 $table->string("min_member_position_strata")->nullable();
30 $table->integer("min_partner")->default(0);
31 $table->unsignedInteger("min_partner_position_id")->nullable();
32 $table->string("min_partner_position_strata")->nullable();
33 $table->timestamps();
34 $table->softDeletes();
35 });
36 }
37
38 /**
39 * Reverse the migrations.
40 *
41 * @return void
42 */
43 public function down()
44 {
45 Schema::dropIfExists('mst_research_req_lecturers');
46 }
47 }

```

**Gambar 4.5. 12 Kode program migrasi Prasyarat Dosen Penelitian**

Gambar 4.5.12 adalah kode program yang penulis kerjakan untuk migrasi tabel mst\_research\_req\_lecturers. Terdapat kode program yang berisi kolom-kolom tabel yang diperlukan sesuai dengan *task*.

- o Relasi di Model Master/Requirement/ResearchReqLecturer:
  - belongsTo:
    - Relasi 1:
      - Nama Relasi: rrlecturer\_researchscheme.
      - Kolom Terkait: research\_scheme\_id.
      - Model Target: /Master/ResearchScheme.
    - Relasi 2:
      - Nama Relasi: rrlecturer\_leadposmin.
      - Kolom Terkait: min\_lead\_position\_id.
      - Model Target: /Master/AcademicPosition.
    - Relasi 3:
      - Nama Relasi: rrlecturer\_leadposmax.
      - Kolom Terkait: max\_lead\_position\_id.
      - Model Target: /Master/AcademicPosition.

- Relasi 4:
  - Nama Relasi: rrlecturer\_memberposmin.
  - Kolom Terkait: min\_member\_position\_id.
  - Model Target: /Master/AcademicPosition.
- Relasi 5:
  - Nama Relasi: rrlecturer\_memberposmax.
  - Kolom Terkait: max\_member\_position\_id.
  - Model Target: /Master/AcademicPosition.
- Relasi 6:
  - Nama Relasi: rrlecturer\_partnerposmin.
  - Kolom Terkait: min\_partner\_position\_id.
  - Model Target: /Master/AcademicPosition.
- Relasi 7:
  - Nama Relasi: rrlecturer\_partnerposmax.
  - Kolom Terkait: max\_partner\_position\_id.
  - Model Target: /Master/AcademicPosition.

**Gambar 4.5. 13 Task model Prasyarat  
Dosen Penelitian**

Gambar 4.5.13 adalah *task* untuk membuat model Prasyarat Dosen Penelitian. Pada *task* ini, penulis membuat model untuk tabel mst\_research\_req\_lecturers bernama ResearchReqLecturer.

```

1 <?php
2
3 namespace App\Models\Master\Requirement;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\SoftDeletes;
8 use App\Models\Master\ResearchScheme;
9 use App\Models\Master\AcademicPosition;
10
11 class ResearchReqLecturer extends Model
12 {
13 use Hasfactory;
14 use SoftDeletes;
15 protected $fillable = [
16 'research_scheme_id',
17 'lecturer_reqs',
18 'min_lecturer',
19 'max_lecturer',
20 'min_lead_position_id',
21 'max_lead_position_id',
22 'min_lead_position_strata',
23 'max_lead_position_strata',
24 'maximum_age',
25 'min_member',
26 'min_member_position_id',
27 'min_member_position_strata',
28 'min_partner',
29 'min_partner_position_id',
30 'min_partner_position_strata'
31];
32 protected $table = "mst_research_req_lecturers";
33 protected $primaryKey = "research_scheme_id";
34 const LECTURER = ['NIDN', 'NIDK', 'NIDW/NIDK'];
35 const STRATA = ['D-I-III', 'S-1', 'S-2', 'S-3', 'PROFESI'];
36 public function rrlecturer_researchscheme(){
37 return $this->belongsTo(ResearchScheme::class,"research_scheme_id");
38 }
39 public function rrlecturer_leadposmin(){
40 return $this->belongsTo(AcademicPosition::class,"min_lead_position_id");
41 }
42 public function rrlecturer_leadposmax(){
43 return $this->belongsTo(AcademicPosition::class,"max_lead_position_id");
44 }
45 public function rrlecturer_memberposmin(){
46 return $this->belongsTo(AcademicPosition::class,"min_member_position_id");
47 }
48 public function rrlecturer_memberposmax(){
49 return $this->belongsTo(AcademicPosition::class,"max_member_position_id");
50 }
51 public function rrlecturer_partnerposmin(){
52 return $this->belongsTo(AcademicPosition::class,"min_partner_position_id");
53 }
54 public function rrlecturer_partnerposmax(){
55 return $this->belongsTo(AcademicPosition::class,"max_partner_position_id");
56 }
57 }

```

**Gambar 4.5. 14 Kode program model Prasyarat Dosen Penelitian**

Gambar 4.5.14 adalah kode program yang penulis kerjakan untuk model ResearchReqLecturer. Terdapat kode program untuk kolom tabel yang dapat diisi secara manual serta relasi tabel yang diperlukan.

#### 4.5.3 Migrasi dan Model Prasyarat Detail Penelitian

##### Tabel mst\_research\_req\_details

- Nama Migrasi: create\_mst\_research\_req\_details\_table.
- Nama Tabel: mst\_research\_req\_details.
- Rincian Kolom:
  - research\_scheme\_id => unsigned integer => Primary Key => FK ke tabel mst\_research\_schemes.
  - min\_mandatory\_output => integer => default 0.
  - min\_student => integer => default 0.
  - max\_student => integer => nullable.
  - disciplines => string => nullable.
  - student\_transport\_cost => float => nullable.
  - student\_transport\_period => string => nullable.
  - waiting\_period => integer => nullable.
  - waiting\_span => string => nullable.
  - timestamps()
  - softDeletes()

**Gambar 4.5. 15 Task migrasi Prasyarat Detail Penelitian**

Gambar 4.5.15 adalah *task* untuk membuat migrasi Prasyarat Detail Penelitian. Pada *task* ini, penulis membuat migrasi untuk tabel mst\_research\_req\_details.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMstBookReqDetailsTable extends Migration
8 {
9 /**
10 * Run the migrations.
11 *
12 * @return void
13 */
14 public function up()
15 {
16 Schema::create('mst_book_req_details', function (Blueprint $table) {
17 $table->unsignedInteger('book_scheme_id')->primary();
18 $table->foreign('book_scheme_id')->references('id')->on('mst_book_schemes');
19 $table->integer('min_lecturer')->default(0);
20 $table->integer('max_lecturer')->nullable();
21 $table->integer('min_mandatory_output')->default(0);
22 $table->integer('max_mandatory_output')->nullable();
23 $table->string('duration_type');
24 $table->integer('max_duration');
25 $table->string('discipline')->nullable();
26 $table->string('book_size');
27 $table->integer('collected_print');
28 $table->boolean('revision')->default(false);
29 $table->timestamps();
30 $table->softDeletes();
31 });
32 }
33
34 /**
35 * Reverse the migrations.
36 *
37 * @return void
38 */
39 public function down()
40 {
41 Schema::dropIfExists('mst_book_req_details');
42 }
43 }
44

```

**Gambar 4.5. 16 Kode Program migrasi Prasyarat Detail Penelitian**

Gambar 4.5.16 adalah kode program yang penulis kerjakan untuk migrasi tabel mst\_research\_req\_details. Terdapat kode program yang berisi kolom-kolom tabel yang diperlukan sesuai dengan *task*.

- Rincian Model:
  - Model: /Master/Requirement/ResearchSchemeDetail.
  - Sertakan const DISCIPLINES = ['MONO', 'MULTI', 'MONO / MULTI'].
  - Sertakan const TRANSPORT\_PERIOD = ['hari', 'minggu', 'bulan', 'semester', 'tahun', 'penelitian'].
  - Sertakan const WAITING\_SPAN = ['hari', 'minggu', 'bulan', 'tahun'].
  - Relasi di Model /Master/Requirement/ResearchReqDetail:
    - belongsTo:
      - Nama Relasi: rrdetail\_researchscheme.
      - Kolom Terkait: research\_scheme\_id.
      - Model Target: /Master/ResearchScheme.

**Gambar 4.5. 17 Task Model Prasyarat Detail Penelitian**

Gambar 4.5.17 adalah *task* untuk membuat model Prasyarat Detail Penelitian. Pada *task* ini, penulis membuat model untuk tabel mst\_research\_req\_details bernama ResearchSchemeDetail.

```

1 <?php
2
3 namespace App\Models\Master\Requirement;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\SoftDeletes;
8 use App\Models\Master\ResearchScheme;
9
10 class ResearchReqDetail extends Model
11 {
12 use HasFactory;
13 use SoftDeletes;
14 protected $fillable = [
15 'research_scheme_id',
16 'min_mandatory_output',
17 'min_student',
18 'max_student',
19 'disciplines',
20 'student_transport_cost',
21 'student_transport_period',
22 'waiting_period',
23 'waiting_span'
24];
25 protected $table = "mst_research_req_details";
26 protected $primaryKey = "research_scheme_id";
27 const DISCIPLINES = ['MONO', 'MULTI', 'MONO / MULTI'];
28 const TRANSPORT_PERIOD = ['hari', 'minggu', 'bulan', 'semester', 'tahun', 'penelitian'];
29 const WAITING_SPAN = ['hari', 'minggu', 'bulan', 'tahun'];
30 public function rrdetail_researchscheme()
31 {
32 return $this->belongsTo(ResearchScheme::class, "research_scheme_id");
33 }
34 }
35

```

**Gambar 4.5. 18 Kode Program Model Prasyarat Detail Penelitian**

Gambar 4.5.18 adalah kode program yang penulis kerjakan untuk model ResearchSchemeDetail. Terdapat kode program untuk kolom tabel yang dapat diisi secara manual serta relasi tabel yang diperlukan.

#### 4.5.4 Migrasi dan Model Prasyarat Anggaran Penelitian

##### Tabel mst\_research\_req\_budgets

- Nama Migrasi: create\_mst\_research\_req\_budgets\_table.
- Nama Tabel: mst\_research\_req\_mandatory\_output.
- Rincian Kolom:
  - research\_scheme\_id => unsigned integer => Primary Key => FK ke tabel mst\_research\_schemes.
  - min\_budget => float => default 0
  - max\_budget => float => nullable
  - lead\_ejm => integer => nullable
  - member\_ejm => integer => nullable
  - ejm\_period => integer => nullable
  - ejm\_period\_type => string => nullable
  - timestamps()
  - softDeletes()

**Gambar 4.5. 19 Task migrasi Prasyarat Anggaran Penelitian**

Gambar 4.5.19 adalah *task* untuk membuat migrasi Prasyarat Anggaran Penelitian. Pada *task* ini, penulis membuat migrasi untuk tabel mst\_research\_req\_mandatory\_output.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMstResearchReqBudgetsTable extends Migration
8 {
9 /**
10 * Run the migrations.
11 *
12 * @return void
13 */
14 public function up()
15 {
16 Schema::create('mst_research_req_mandatory_output', function (Blueprint $table) {
17 $table->unsignedInteger("research_scheme_id")->primary();
18 $table->foreign("research_scheme_id")->references("id")->on("mst_research_schemes");
19 $table->float("min_budget")->default(0);
20 $table->float("max_budget")->nullable();
21 $table->integer("lead_ejm")->nullable();
22 $table->integer("member_ejm")->nullable();
23 $table->integer("ejm_period")->nullable();
24 $table->string("ejm_period_type")->nullable();
25 $table->timestamps();
26 $table->softDeletes();
27 });
28 }
29
30 /**
31 * Reverse the migrations.
32 *
33 * @return void
34 */
35 public function down()
36 {
37 Schema::dropIfExists('mst_research_req_mandatory_output');
38 }
39 }

```

**Gambar 4.5. 20 Kode Program migrasi Prasyarat Anggaran Penelitian**

Gambar 4.5.20 adalah kode program yang penulis kerjakan untuk migrasi tabel mst\_research\_mandatory\_output. Terdapat kode program yang berisi kolom-kolom tabel yang diperlukan sesuai dengan *task*.

- Rincian Model:
  - Model: Master/Requirement/ResearchReqBudget.
  - Sertakan const TYPE = ['semester', 'tahun', 'periode'].
  - Relasi di Model Master/Requirement/ResearchReqBudget:
    - belongsTo:
      - Nama Relasi: rrbudget\_researchscheme.
      - Kolom Terkait: research\_scheme\_id.
      - Model Target: /Master/ResearchScheme.

**Gambar 4.5. 21 Task model Prasyarat Anggaran Penelitian**

Gambar 4.5.21 adalah *task* untuk membuat model Prasyarat Anggaran Penelitian. Pada *task* ini, penulis membuat model untuk tabel mst\_research\_req\_mandatory\_output bernama ResearchReqBudget.

```

1 <?php
2
3 namespace App\Models\Master\Requirement;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\SoftDeletes;
8 use App\Models\Master\ResearchScheme;
9
10 class ResearchReqBudget extends Model
11 {
12 use HasFactory;
13 protected $fillable = [
14 'research_scheme_id',
15 'min_budget',
16 'max_budget',
17 'lead_ejm',
18 'member_ejm',
19 'ejm_period',
20 'ejm_period_type',
21];
22 protected $table = "mst_research_req_mandatory_output";
23 protected $primaryKey = "research_scheme_id";
24 const TYPE = ['semester', 'tahun', 'periode'];
25 public function rrbudget_researchscheme(){
26 return $this->belongsTo(ResearchScheme::class,"research_scheme_id");
27 }
28 }
29

```

**Gambar 4.5. 22 Kode Program model Prasyarat Anggaran Penelitian**

Gambar 4.5.22 adalah kode program yang penulis kerjakan untuk model ResearchReqBudget. Terdapat kode program untuk kolom tabel yang dapat diisi secara manual serta relasi tabel yang diperlukan.

#### 4.5.5 Migrasi dan Model Prasyarat Program Penelitian

##### Tabel mst\_research\_req\_programmes

- Nama Migrasi: create\_mst\_research\_req\_programmes\_table.
- Nama Tabel: mst\_research\_req\_programmes.
- Rincian Kolom:
  - research\_scheme\_id => unsigned integer => FK ke tabel mst\_research\_schemes.
  - student\_programme\_id => unsigned integer => FK ke tabel mst\_student\_programmes.



- timestamps()
- softDeletes()
- Set kombinasi antara kolom research\_scheme\_id dgn kolom student\_programme\_id sebagai Composite Primary Key (utk relasi many to many).

**Gambar 4.5. 23 Task migrasi Prasyarat Program Penelitian**

Gambar 4.5.23 adalah *task* untuk membuat migrasi Prasyarat Program Penelitian. Pada *task* ini, penulis membuat migrasi untuk tabel mst\_research\_req\_programmes. Tabel ini sebenarnya adalah tabel yang terbentuk karena tabel Skema Penelitian memiliki relasi *many to many* dengan tabel Program Mahasiswa.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMstResearchReqProgrammesTable extends Migration
8 {
9 /**
10 * Run the migrations.
11 *
12 * @return void
13 */
14 public function up()
15 {
16 Schema::create('mst_research_req_programmes', function (Blueprint $table) {
17 $table->unsignedInteger("research_scheme_id")->index();
18 $table->foreign("research_scheme_id")->references("id")->on("mst_research_schemes");
19 $table->unsignedInteger("student_programme_id")->index();
20 $table->foreign("student_programme_id")->references("id")->on("mst_student_programmes");
21 $table->timestamps();
22 $table->softDeletes();
23 $table->primary(["research_scheme_id", "student_programme_id"]);
24 });
25 }
26
27 /**
28 * Reverse the migrations.
29 *
30 * @return void
31 */
32 public function down()
33 {
34 Schema::dropIfExists('mst_research_req_programmes');
35 }
36 }
37

```

Gambar 4.5. 24 Kode program migrasi Prasyarat Program Penelitian

Gambar 4.5.24 adalah kode program yang penulis kerjakan untuk migrasi tabel mst\_research\_req\_programmes. Terdapat kode program yang berisi kolom-kolom tabel yang diperlukan sesuai dengan *task*. Untuk kolom research\_scheme\_id dan student\_programme\_id dibuat menjadi *composite primary key* karena tabel mst\_research\_req\_programmes dan tabel mst\_student\_programmes memiliki relasi *many to many*.

- Rincian Model:
  - Relasi di Model Master/ResearchScheme:
    - belongsToMany:
    - Nama Relasi: researchscheme\_studentprogramme.
    - Kolom Terkait: research\_scheme\_id.
    - Model Target: Master/StudentProgramme.
  - Relasi di Model Master/StudentProgramme:
    - belongsToMany:
    - Nama Relasi: studentprogramme\_researchscheme.
    - Kolom Terkait: student\_programme\_id.
    - Model Target: Master/ResearchScheme.

**Gambar 4.5. 25 Task model Prasyarat Program Penelitian**

Gambar 4.5.25 adalah *task* untuk membuat relasi tabel Prasyarat Program Penelitian. Pada *task* ini, penulis membuat relasi pada model ResearchScheme dan StudentProgramme.

```
17 | public function studentprogramme_researchscheme(){
18 | return $this->belongsToMany(ResearchScheme::class,"mst_research_req_programmes","student_programme_id","research_scheme_id");
19 | }
```

**Gambar 4.5. 26 kode program di model StudentProgramme**

Gambar 4.5.26 adalah kode program yang telah penulis kerjakan untuk membuat relasi antara tabel mst\_student\_programme dengan tabel mst\_research\_scheme di model StudentProgramme.

```
31 | public function researchscheme_studentprogramme(){
32 | return $this->belongsToMany(StudentProgramme::class,"mst_research_req_programmes","research_scheme_id","student_programme_id");
33 | }
```

**Gambar 4.5. 27 kode program di model ResearchScheme**

Gambar 4.5.27 adalah kode program yang telah penulis kerjakan untuk membuat relasi antara tabel mst\_research\_scheme dengan tabel mst\_student\_programme di model ResearchScheme.

#### 4.5.6 Migrasi dan Model Prasyarat Luaran Penelitian

##### Tabel mst\_research\_req\_outputs

- Nama Migrasi: create\_mst\_research\_req\_outputs\_table.
- Nama Tabel: mst\_research\_req\_outputs.
- Rincian Kolom:
  - id
  - research\_scheme\_id => unsigned integer => FK ke tabel mst\_research\_schemes.
  - output\_id => unsigned integer => FK ke tabel mst\_outputs.
  - category => string
  - output\_title => string
  - min\_grade => string => nullable
  - max\_grade => string => nullable
  - timestamps()
  - softDeletes()

Gambar 4.5. 28 Task migrasi Prasyarat Luaran Penelitian

Gambar 4.5.28 adalah *task* untuk membuat migrasi Prasyarat Luaran Penelitian. Pada *task* ini, penulis membuat migrasi untuk tabel mst\_research\_req\_outputs.

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMstResearchReqOutputsTable extends Migration
8 {
9 /**
10 * Run the migrations.
11 *
12 * @return void
13 */
14 public function up()
15 {
16 Schema::create('mst_research_req_outputs', function (Blueprint $table) {
17 $table->id();
18 $table->unsignedInteger("research_scheme_id");
19 $table->foreign("research_scheme_id")->references("id")->on("mst_research_schemes");
20 $table->unsignedInteger("output_id");
21 $table->foreign("output_id")->references("id")->on("mst_outputs");
22 $table->string("category");
23 $table->string("output_title");
24 $table->string("min_grade")->nullable();
25 $table->string("max_grade")->nullable();
26 $table->timestamps();
27 $table->softDeletes();
28 });
29 }
30
31 /**
32 * Reverse the migrations.
33 *
34 * @return void
35 */
36 public function down()
37 {
38 Schema::dropIfExists('mst_research_req_outputs');
39 }
40 }
41

```

Gambar 4.5. 29 Kode Program migrasi Prasyarat Luaran Penelitian

Gambar 4.5.29 adalah kode program yang penulis kerjakan untuk migrasi tabel mst\_research\_req\_outputs. Terdapat kode program yang berisi kolom-kolom tabel yang diperlukan sesuai dengan *task*.

- Rincian Model:
  - Model: Master/Requirement/ResearchReqOutput.
  - Sertakan const CATEGORY = ['wajib', 'tambahan'].
  - Relasi di Model Master/Requirement/ResearchReqOutput:
    - belongsTo:
      - Relasi 1:
        - Nama Relasi: rroutput\_researchscheme.
        - Kolom Terkait: research\_scheme\_id.
        - Model Target: /Master/ResearchScheme.
      - Relasi 2:
        - Nama Relasi: rroutput\_output.
        - Kolom Terkait: output\_id.
        - Model Target: /Master/Output.

**Gambar 4.5. 30 Task model Prasyarat Luaran Penelitian**

Gambar 4.5.30 adalah *task* untuk membuat model Prasyarat Luaran Penelitian. Pada task ini, penulis membuat model untuk tabel mst\_research\_req\_outputs bernama ResearchReqOutput.

```

1 <?php
2
3 namespace App\Models\Master\Requirement;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\SoftDeletes;
7 use Illuminate\Database\Eloquent\Model;
8 use App\Models\Master\ResearchScheme;
9 use App\Models\Master\Output;
10
11 class ResearchReqOutput extends Model
12 {
13 use HasFactory;
14 use SoftDeletes;
15 protected $fillable = [
16 'research_scheme_id',
17 'output_id',
18 'category',
19 'output_title',
20 'min_grade',
21 'max_grade'
22];
23 protected $table = "mst_research_req_outputs";
24 const CATEGORY = ['wajib', 'tambahan'];
25 public function rroutput_researchscheme(){
26 return $this->belongsTo(ResearchScheme::class,"research_scheme_id");
27 }
28 public function rroutput_output(){
29 return $this->belongsTo(Output::class,"output_id");
30 }
31 }
32
```

**Gambar 4.5. 31 Kode Program model Prasyarat Luaran Penelitian**

Gambar 4.5.29 adalah kode program yang penulis kerjakan untuk model ResearchReqOutput. Terdapat kode program untuk kolom tabel yang dapat diisi secara manual serta relasi tabel yang diperlukan.

Setelah selesai mengerjakan untuk migrasi dan model modul penelitian, penulis kemudian melakukan *push* melalui Github Desktop.

## 4.6 Prasyarat Program Penelitian

Pada modul penelitian terdapat beberapa prasyarat yang harus dipenuhi agar penelitian tersebut disetujui. Pada proyek SIPP ini, prasyarat tersebut diantaranya ada Prasyarat Detail Penelitian, Prasyarat Program Penelitian, Prasyarat Luaran Penelitian, Prasyarat Anggaran Penelitian, dan Prasyarat Dosen Penelitian. Penulis mendapat *task* untuk mengerjakan Prasyarat Program Penelitian. Pada bagian ini, penulis membuat *create, read, delete* (CRD) untuk Prasyarat Program Penelitian. Sebelum membuat kode program untuk CRD Prasyarat Program Penelitian, penulis melakukan *pull* melalui Github Desktop terlebih dahulu. Kemudian penulis melakukan *import* untuk DataTables, SWAL, dan Select2.

```

1 @extends('dashboard.base')
2
3 @section('css')
4 {{-- Datatables --}}
5 <meta name="csrf-token" content="{!! csrf_token() !!}" />
6 <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.21/css/dataTables.bootstrap4.min.css">
7 <style>
8 .dataTables_wrapper select {
9 min-width: 50px !important;
10 }
11
12 #program {
13 width: 200px;
14 }
15 </style>
16
17 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
18 <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
19 <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
20
21 @endsection

```

Gambar 4.6. 1 Kode program *import* CSS DataTables, dan Select2

```

106 <script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.21/js/jquery.dataTables.min.js">
107 </script>
108 <script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.21/js/dataTables.bootstrap4.min.js">
109 </script>
110 <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

```

Gambar 4.6. 2 Kode program *import* JavaScript DataTables dan SWAL

Gambar 4.6.1 dan gambar 4.6.2 adalah kode program yang penulis kerjakan untuk melakukan *import/install* CDN untuk DataTables (CSS dan JavaScript),

Select2, dan Swal. Kemudian terdapat juga kode program agar DataTables, Select2, dan Swal dapat dipakai.

```

112 <script>
113 $(document).ready(function() {
114 $('#spTable').DataTable();
115 $('#program').select2();
116 });
117
118 $('.btn-delete').click(function() {
119 var dt = $(this).data('dt');
120 Swal.fire([
121 title: "Yakin mau menghapus Program Mahasiswa ini?",
122 icon: "warning",
123 confirmButtonText: "Hapus",
124 cancelButtonText: "Kembali",
125 confirmButtonColor: "#d9534f",
126 showCancelButton: true,
127 allowOutsideClick: false
128]).then((response) => {
129 if (response.isConfirmed) {
130 $('#deleteid').val(dt.id);
131 $('#deleteform').submit();
132 }
133 });
134 });
135 </script>
```

**Gambar 4.6.3 Kode program JavaScript DataTables, Select2, dan Swal**

Gambar 4.6.3 adalah kode program yang penulis kerjakan agar DataTables, Select2, dan Swal dapat digunakan.

#### 4.6.1 Read

##### Prasyarat Program Penelitian

###### Main Page

- Group Route: admin | lppm
- Route: /master/skema-penelitian/{research\_scheme\_id}/requirement/program
- Controller: Master/ResearchSchemeController@index\_research\_sp\_req
- Method: GET
- View: dashboard/master/requirement/research-req-sp
- Rincian Tampilan:
  - Tampilan ingin dibuat seperti halaman master data pada umumnya:

| # | Nama                   | Aksi            |
|---|------------------------|-----------------|
| 1 | Rektor                 | [Edit] [Delete] |
| 2 | Menteri                | [Edit] [Delete] |
| 3 | Pengaruh Internasional | [Edit] [Delete] |
| 4 | Orang Saya             | [Edit] [Delete] |

- Judul: "Prasyarat Program Penelitian: (inst\_research\_schemes->research\_scheme)".
- Tombol (yg sejajar judul):
  - Tombol "Kembali". Link ke halaman /master/skema-penelitian/{research\_scheme\_id}/requirement.
  - Tombol "Tambah". Lihat **"Insert Prasyarat Program Penelitian"**
- Tabel menggunakan datatable:

```
// Pada section css tambahkan:
<link rel="stylesheet" type="text/css"
 href="https://cdn.datatables.net/1.10.21/css/dataTables.bootstrap4.min.css">

// Pada section javascript tambahkan:
<script type="text/javascript" charset="utf8"
 src="https://cdn.datatables.net/1.10.21/js/jquery.dataTables.min.js">
```

- Data diambil dari tabel mst\_research\_req\_programmes dimana research\_scheme\_id = {research\_scheme\_id}:
 

| # | Program Mahasiswa | Aksi |
|---|-------------------|------|
|   |                   |      |
|   |                   |      |
|   |                   |      |

- 
- Kolom "Program Mahasiswa": mst\_student\_programmes -> programme\_name
  - Kolom "Aksi":
    - Tombol "Hapus". Lihat bagian **"Delete Prasyarat Program Penelitian"**. Sertakan atribut data-dt = { mst\_student\_programmes }

Gambar 4.6.4 Task read dan main page Prasyarat Program Penelitian

Gambar 4.6.4 adalah task untuk membuat main page pada Prasyarat Program Penelitian.

```

22
23 @section('content')
24 @if ($errors->any())
25 <div class="alert alert-danger">
26 {{ $errors->first() }}
27 </div>
28 @endif
29 @if ($session::has('message'))
30 <div class="alert alert-success">
31 {{ Session::get('message') }}
32 </div>
33 @endif
34 <div class="card">
35 <div class="card-header">
36 <div class="d-flex justify-content-between">
37 <h4 class="mt-1">Prasyarat Program Penelitian: {{ $schemes->research_scheme }}

```

Gambar 4.6. 5 Kode Program *Main page* Prasyarat Program Penelitian

Gambar 4.6.5 adalah kode program yang penulis kerjakan pada *view* untuk membuat *main page* dari Prasyarat Program Penelitian. Terdapat kode program untuk membuat bagian judul, tombol “Kembali” untuk menuju halaman Prasyarat Penelitian, tombol “Tambah” untuk memunculkan modal yang nantinya berfungsi untuk menambah Program Mahasiswa. Kemudian terdapat kode program untuk membuat bagian tabel menggunakan DataTable. Tabel tersebut menggunakan id “spTable” sesuai dengan jQuery. Tabel tersebut berisi 3 kolom, kolom pertama berupa nomor urut, kolom kedua berisi Program Mahasiswa, dan kolom ketiga berisi button “Delete” untuk menghapus Program Mahasiswa tersebut dari Prasyarat Program Penelitian.

```
Route::get('skema-penelitian/{research_scheme_id}/requirement/program', [ResearchSchemeController::class, "index_research_sp_req"]);
```

Gambar 4.6. 6 Kode program *route read* Prasyarat Program Penelitian

Gambar 4.6.6 adalah kode program yang penulis kerjakan untuk membuat *route read* untuk Prasyarat Program Penelitian. *Route* ini nantinya akan menjadi alamat dari halaman web Prasyarat Program Penelitian. Metode yang digunakan adalah *get*, karena berfungsi untuk menampilkan data. *Controller* yang dipakai adalah ResearchSchemeController dengan fungsi bernama “index\_research\_req\_sp”.

```

public function index_research_sp_req($research_scheme_id)
{
 $schemes = ResearchScheme::find($research_scheme_id);
 $reqprograms = $schemes->researchscheme_studentprogramme;
 $programs = StudentProgramme::all();
 $programcoll = [];
 foreach ($programs as $item) {
 $programcoll[$item->id] = $item->programme_name;
 }
 return view('dashboard.master.requirement.research-req-sp', compact("programcoll", "schemes", "reqprograms"));
}

```

Gambar 4.6.7 Kode program *controller* read Prasyarat Program Penelitian

Gambar 4.6.7 adalah kode program yang penulis kerjakan pada *controller* untuk menampilkan data Program Mahasiswa yang menjadi prasyarat dari Prasyarat Program Penelitian. Nama fungsinya adalah “index\_research\_req\_sp”. Pada fungsi ini terdapat kode program yang berfungsi untuk mencari/find id dari Program Penelitian berdasarkan parameter. Lalu menampilkan Program Mahasiswa yang menjadi prasyarat dari Program Penelitian tersebut. Pada fungsi ini juga terdapat kode program untuk menampilkan data-data Program Mahasiswa yang nantinya akan ditampilkan pada modal ketika ingin menambahkan prasyarat.

Tabel mst\_research\_req\_programmes saat ini masih kosong atau belum ada data, sehingga penulis perlu mengisi data *dummy* ke dalam tabel tersebut. Untuk mengisi data *dummy* ke dalam tabel mst\_research\_req\_programmes, penulis menggunakan salah satu fitur dari Laravel yaitu *seeder*.

```

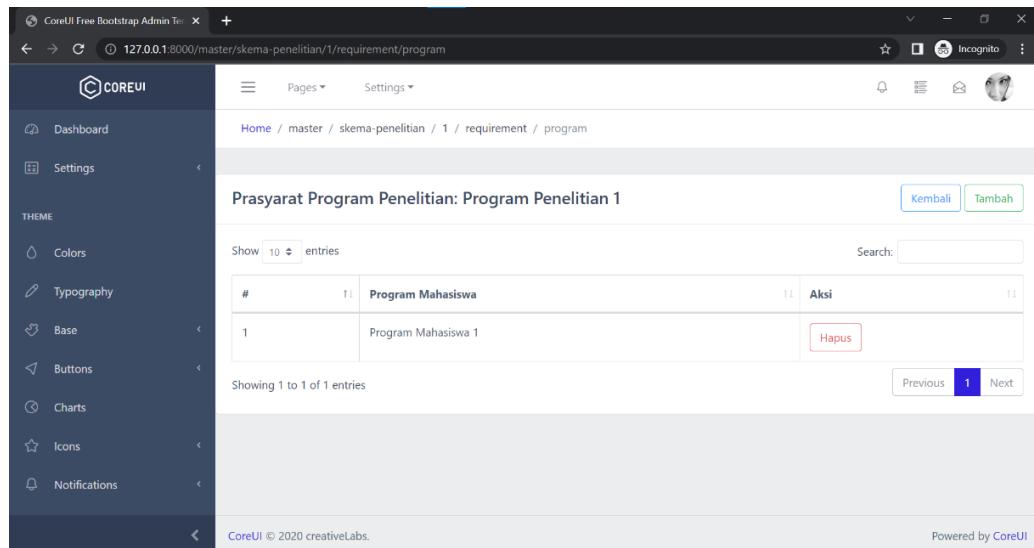
public function run()
{
 DB::table('mst_student_programmes')->insert([
 "programme_name"=>"Program Mahasiswa 1",
]);
 DB::table('mst_student_programmes')->insert([
 "programme_name"=>"Program Mahasiswa 2",
]);
 DB::table('mst_student_programmes')->insert([
 "programme_name"=>"Program Mahasiswa 3",
]);
 DB::table('mst_research_schemes')->insert([
 "research_scheme"=>"Program Penelitian 1",
]);
 DB::table('mst_research_req_programmes')->insert([
 "research_scheme_id"=>1,
 "student_programme_id"=>1
]);
}

```

Gambar 4.6.8 Kode program untuk *seeder*

Gambar 4.6.8 adalah kode program yang penulis kerjakan pada bagian *seeder*. Pada kode program tersebut, penulis menambahkan data pada tabel mst\_student\_programmes, mst\_research\_schemes, dan mst\_research\_programmes. Untuk tabel mst\_student\_programmes, penulis memasukkan data untuk kolom programme\_name, di mana nantinya data pada

kolom ini menjadi Program Mahasiswa.Untuk kolom id tidak perlu diisi karena sudah *autoincrement*. Untuk tabel mst\_research\_schemes, penulis memasukkan data untuk kolom research\_scheme, di mana nantinya data pada kolom ini akan menjadi data Program Penelitian. Untuk kolom id tidak perlu diisi karena sudah *autoincrement*. Untuk tabel mst\_research\_programmes, penulis memasukkan data pada 2 kolom, yaitu kolom research\_scheme\_id diisi dengan id 1 dan kolom student\_programme\_id diisi dengan id 1. Dengan memasukkan data ke dalam tabel-tabel tersebut, nantinya tabel pada *main page* sudah dapat terisi, serta pilihan pada *select* di modal pada saat akan melakukan penambahan prasyarat pun sudah terisi. Kemudian pada Command Prompt penulis harus mengetik kode perintah untuk menejalankan *seeder* tersebut.



Gambar 4.6.9 Tampilan *main page* dan *read* Prasyarat Program Penelitian

Gambar 4.6.9 merupakan tampilan Prasyarat Program Penelitian dari Program Penelitian 1. Pada tampilan tersebut terdapat tabel yang menampilkan Program Mahasiswa yang menjadi prasyarat dalam Program Penelitian 1. Tabel tersebut menggunakan fitur DataTables. Terdapat juga tombol “Tambah” untuk menambahkan Program Mahasiswa serta tombol “Hapus” untuk menghapus Program Mahasiswa tersebut. Selain itu, terdapat juga penggunaan *template* CoreUI.

## 4.6.2 *Create*

### **Insert Prasyarat Program Penelitian**

- Ketika tombol di-klik, maka munculkan Modal "Tambah Prasyarat Program Penelitian" yg berisikan form dgn fields sbb:
  - Input select "Program Mahasiswa". Diambil dari tabel mst\_student\_programmes.
    - Value = id
    - Text = programme\_name
    - Terapkan select2. Ref: <https://select2.org/>
    - Multiple selection
  - Pada Modal Footer:
    - Tombol "Batal". Ketika di-klik, maka close Modal.
    - Tombol "Simpan". Ketika di-klik, maka submit form dan INSERT data ke database.
- INSERT data ke database:
  - Grup Route: admin|lppm
  - Route: /master/skema-penelitian/{research\_scheme\_id}/requirement/program
  - Controller: Master/ResearchSchemeController@store\_research\_req\_sp
  - Method: POST
  - Redirect /master/skema-penelitian/{research\_scheme\_id}/requirement/program dgn message:
    - Jika proses insert berhasil: "Prasyarat telah berhasil ditambahkan".
    - Jika proses insert gagal: "{error message dari Controller}".
  - Rincian di Controller:
    - FIND data mst\_research\_scheme = {research\_scheme\_id}
    - Sync data mst\_research\_req\_programmes sesuai dengan input select "Program Mahasiswa".
      - Note: bisa menggunakan teknik sync, maupun attach dan detach.

**Gambar 4.6. 10 Task *create/insert* Prasyarat Program Penelitian**

Gambar 4.6.10 adalah *task* untuk membuat bagian *create* atau *insert* pada Prasyarat Program Penelitian.

```

62 </tr>
63 @endforeach
64 </tbody>
65 </table>
66 <div class="modal fade" tabindex="-1" id="tambahsp">
67 <div class="modal-dialog">
68 <div class="modal-content">
69 <div class="modal-header">
70 <h5 class="modal-title">Tambah Prasyarat Program Penelitian</h5>
71 <button type="button" class="close" data-dismiss="modal"
72 aria-label="Close">×</button>
73 </div>
74 <div class="modal-body">
75 {{ Form::open(['method' => 'POST', 'url' => '/master/skema-penelitian/' . $schemes->id . '/requirement/program']) }}
76 <div class="row">
77 <div class="col-4">
78 {{ Form::label('namaprogram[]', 'Nama Program', ['class' => 'form-label']) }}
79 </div>
80 <div class="col">
81 {{ Form::select('namaprogram[]', $programcoll,null,['id'=>"program", "multiple" => "multiple"]) }}
82 </div>
83 </div>
84 </div>
85 <div class="modal-footer">
86 {{ Form::reset('Batal', ['class' => 'btn btn-outline-secondary', 'data-dismiss' => 'modal']) }}
87 {{ Form::submit('Simpan', ['class' => 'btn btn-outline-primary']) }}
88 </div>
89 {{ Form::close() }}
90 </div>
91 </div>
92 </div>
93 </div>
94

```

Gambar 4.6. 11 Kode program *view insert* Prasyarat Program Penelitian

Gambar 4.6.11 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan modal untuk menambah Program Mahasiswa sebagai prasyarat untuk Prasyarat Program Penelitian. Terdapat kode program untuk membuat modal dengan judul “Tambah Prasyarat Program Penelitian” di mana isi dari modal tersebut terdapat label, *select*, tombol “Batal”, dan tombol “Simpan”. Untuk label akan memunculkan tulisan “Nama Program”. Untuk *select* menggunakan fitur dari *Select2* dengan *selection multiple* serta menggunakan id “program” sesuai dengan jQuery. Isi dari *select* ini adalah data-data Program Mahasiswa. Tombol “Batal” untuk membatalkan penambahan prasyarat Program Penelitian. Tombol “Simpan” untuk menambah/menyimpan Program Mahasiswa yang telah dipilih dari *select*.

```
Route::post('skema-penelitian/{research_scheme_id}/requirement/program', [ResearchSchemeController::class, "store_research_req_sp"]);
```

Gambar 4.6. 12 Kode program *route create* Prasyarat Program Penelitian

Gambar 4.6.12 adalah kode program yang penulis kerjakan untuk membuat *route create* pada Prasyarat Program Penelitian. *Route* ini nantinya akan menjadi alamat dari halaman web Prasyarat Program Penelitian. Metode yang digunakan adalah *post*, karena berfungsi untuk menyimpan/menambah data. *Controller* yang dipakai adalah ResearchSchemeController dengan fungsi bernama “store\_research\_req\_sp”.

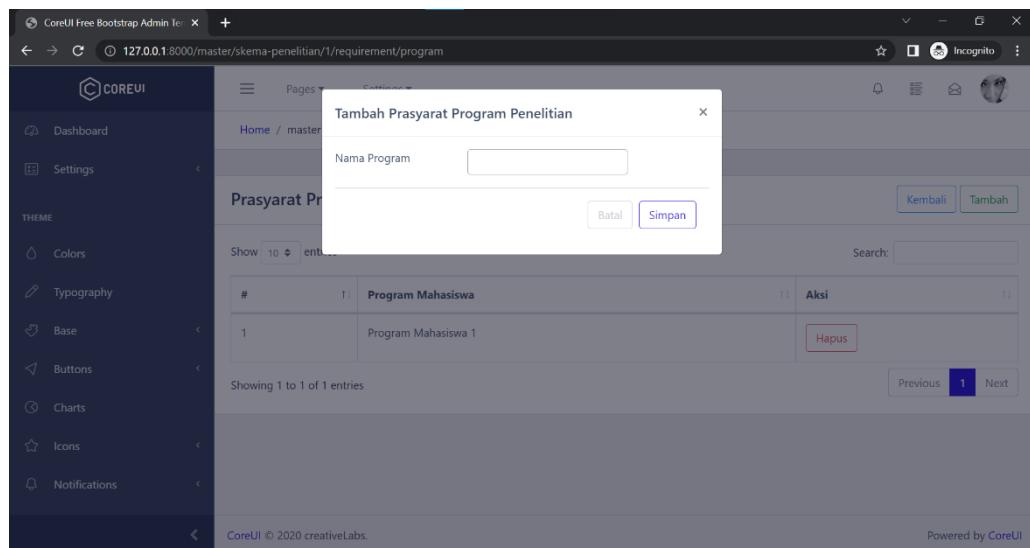
```

261 public function store_research_req_sp(Request $request, $research_scheme_id)
262 {
263 $schemes = ResearchScheme::find($research_scheme_id);
264 if($request->namaprogram == null){
265 return back()->withErrors("Gagal menambahkan Prasyarat");
266 }
267 try {
268 $schemes->researchscheme_studentprogramme()->attach($request->namaprogram);
269 } catch (QueryException $e) {
270 return back()->withErrors("Gagal menambahkan Prasyarat");
271 }
272
273 return redirect("/master/skema-penelitian/" . $research_scheme_id . "/requirement/program")->with("message", "Prasyarat telah berhasil ditambahkan");
274 }

```

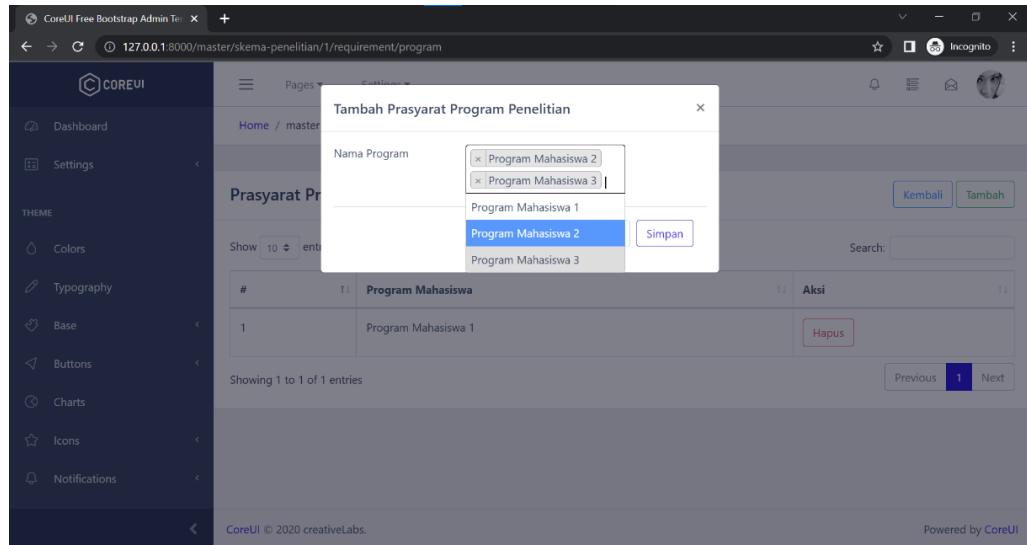
Gambar 4.6. 13 Kode program *controller store* Prasyarat Program Penelitian

Gambar 4.6.13 adalah kode program yang penulis kerjakan pada *controller* untuk menambahkan Program Mahasiswa sebagai prasyarat dari Prasyarat Program Penelitian. Nama fungsinya adalah “store\_research\_req\_sp”. Pada fungsi ini terdapat kode program untuk mencari/find id dari Program Penelitian berdasarkan parameter. Jika id tidak ditemukan, maka gagal untuk ditambahkan, lalu akan memunculkan pesan pada *main page* “Gagal menambahkan prasyarat”. Jika id ditemukan, maka Program Mahasiswa tersebut akan ditambahkan ke dalam prasyarat Program Penelitian tersebut dengan teknik *attach* kemudian muncul pesan pada *main page* “Prasyarat telah berhasil ditambahkan”. Terdapat juga kode program untuk mencegah apabila Program Mahasiswa kosong atau tidak diisi maka akan memunculkan pesan pada *main page* bahwa prasyarat tersebut gagal untuk ditambahkan.



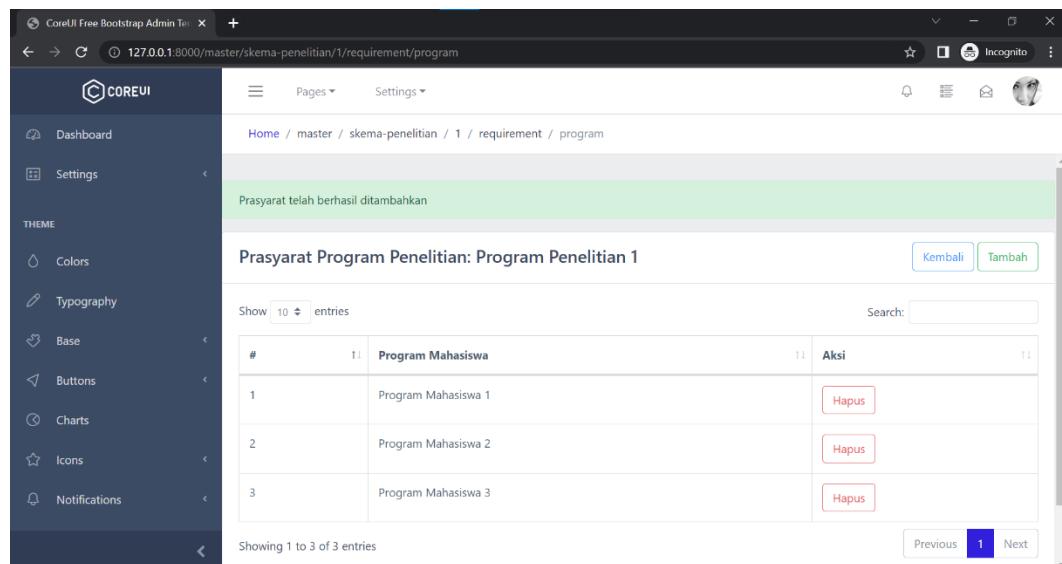
Gambar 4.6. 14 Tampilan modal menambah Prasyarat Program Penelitian

Gambar 4.6.14 merupakan tampilan modal untuk menambahkan Prasyarat Program Penelitian. Tampilan modal ini akan muncul setelah tombol “Tambah” diklik. Modal tersebut menggunakan fitur dari Bootstrap.



Gambar 4.6. 15 Tampilan Select2 dengan *selection multiple*

Gambar 4.6.15 merupakan tampilan dari penerapan Select2 dengan *selection multiple*. *Selection multiple* memungkinkan untuk dapat menambah Program Mahasiswa lebih dari 1 sekaligus. Jika sudah memilih nama program, selanjutnya klik “Simpan” untuk menambahkan Program Mahasiswa tersebut ke dalam prasyarat.



Gambar 4.6. 16 Tampilan Program Mahasiswa berhasil ditambahkan

Gambar 4.6.16 adalah tampilan ketika tombol “Simpan” pada modal diklik dan Program Mahasiswa berhasil ditambahkan ke dalam Prasyarat Program Penelitian. Pada *main page* akan muncul pesan “Prasyarat telah berhasil ditambahkan”

#### 4.6.3 Delete

##### Delete Prasyarat Program Penelitian

- Munculkan SWAL konfirmasi dgn rincian sbb:
  - Tulisan "Yakin mau menghapus Program Mahasiswa ini?".



- Tombol "Kembali". Ketika di-klik, close SWAL.
- Tombol "Hapus". Ketika di-klik, DELETE data pada database.
- Untuk bisa menggunakan SWAL, pada bagian section javascript tambahkan:
 

```
<script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
```

  - Dokumentasi Sweet Alert 2: <https://sweetalert2.github.io/>
- DELETE data pada database:
  - Grup Route: admin|lppm
  - Route: /master/skema-penelitian/{research\_scheme\_id}/requirement/program
  - Controller: Master/ResearchSchemeController@delete\_research\_req\_sp
  - Method: DELETE
  - Redirect: /master/skema-pelayanan-publik dengan message:
    - Jika proses delete berhasil: "Program Mahasiswa telah berhasil dihapus".
    - Jika proses delete gagal: "{error message dari Controller}".
  - Rincian di Controller:
    - FIND data mst\_research\_scheme = {research\_scheme\_id}
    - Detach data mst\_research\_req\_programmes yang memiliki data-dt->id.

**Gambar 4.6. 17 Task delete Prasyarat Program Penelitian**

Gambar 4.6.17 adalah *task* untuk membuat bagian *delete* pada Prasyarat Program Penelitian.

```

97 <div>
98 {{ Form::open(['method' => 'DELETE', 'id' => 'deleteform']) }}
99 {{ Form::hidden('id', null, ['id' => 'deleteid']) }}
100 </div>
101 @endsection
102
103 @section('javascript')
104
105 <script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.21/js/jquery.dataTables.min.js">
106 </script>
107 <script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.21/js/dataTables.bootstrap4.min.js">
108 </script>
109 <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
110
111 <script>
112 $(document).ready(function() {
113 $('#spTable').DataTable();
114 $('#program').select2();
115 });
116
117 $('.btn-delete').click(function() {
118 var dt = $(this).data('dt');
119 Swal.fire({
120 title: "Yakin mau menghapus Program Mahasiswa ini?",
121 icon: "warning",
122 confirmButtonText: "Hapus",
123 cancelButtonText: "Kembali",
124 confirmButtonColor: "#d9534f",
125 showCancelButton: true,
126 allowOutsideClick: false
127 }).then((response) => {
128 if (response.isConfirmed) {
129 $('#deleteid').val(dt.id);
130 $('#deleteform').submit();
131 }
132 });
133 });
134 </script>
135
```

Gambar 4.6. 18 Kode Program *view delete* Prasyarat Program Penelitian

Gambar 4.6.18 adalah kode program yang penulis kerjakan pada *view* untuk membuat tampilan ketika ingin menghapus Program Mahasiswa yang dipilih. Untuk tampilannya menggunakan fitur dari SWAL (*Sweet Alert*). Terdapat kode program untuk membuat form delete. Kemudian terdapat kode program untuk penggunaan SWAL. Pada SWAL ini terdapat pesan “Yakin mau menghapus Program Mahasiswa ini?”, tombol “Hapus” untuk menghapus Program Mahasiswa tersebut dari prasyarat, serta tombol “Kembali” untuk kembali ke halaman *main page*.

```
Route::delete('skema-penelitian/{research_scheme_id}/requirement/program', [ResearchSchemeController::class, "delete_research_req_sp"]);
```

Gambar 4.6. 19 Kode program *route delete* Prasyarat Program Penelitian

Gambar 4.6.19 adalah kode program yang penulis kerjakan untuk membuat *route delete* pada Prasyarat Program Penelitian. *Route* ini nantinya akan menjadi alamat dari halaman web Prasyarat Program Penelitian. Metode yang digunakan adalah *delete*, karena berfungsi untuk menghapus data. *Controller* yang dipakai adalah ResearchSchemeController dengan fungsi bernama “*delete\_research\_req\_sp*”.

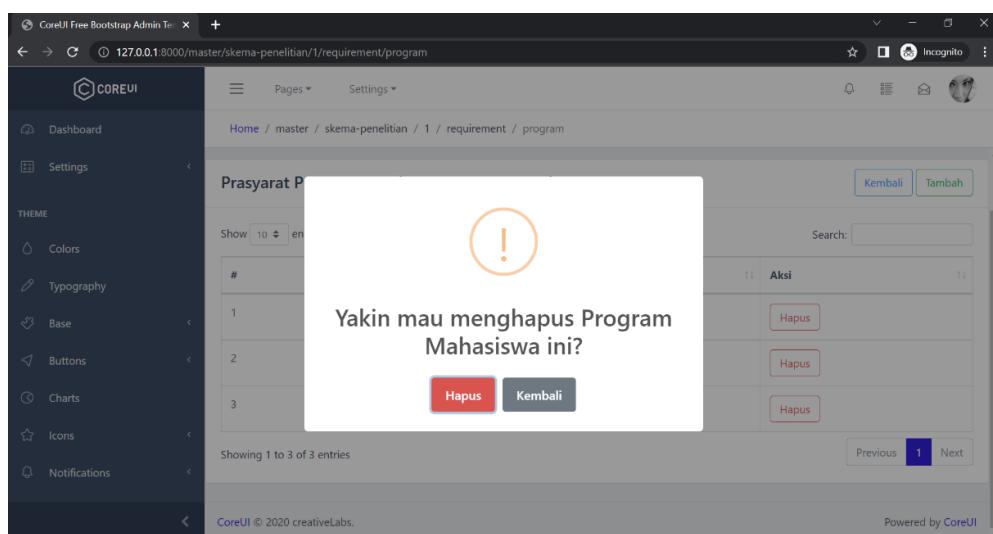
```

274 | public function delete_research_req_sp(Request $request, $research_scheme_id)
275 |
276 | {
277 | $schemes = ResearchScheme::find($research_scheme_id);
278 | if ($schemes == null) {
279 | return back()->withErrors(['status' => "Data tidak ditemukan"]);
280 | }
281 | try {
282 | $schemes->researchscheme_studentprogramme()->detach($request->id);
283 | } catch (QueryException $e) {
284 | return back()->withErrors(['status' => "Gagal menghapus Program Mahasiswa"]);
285 | }
286 | return redirect('master/skema-penelitian/' . $research_scheme_id . '/requirement/program')->with("message", "Program Mahasiswa telah berhasil dihapus");
}

```

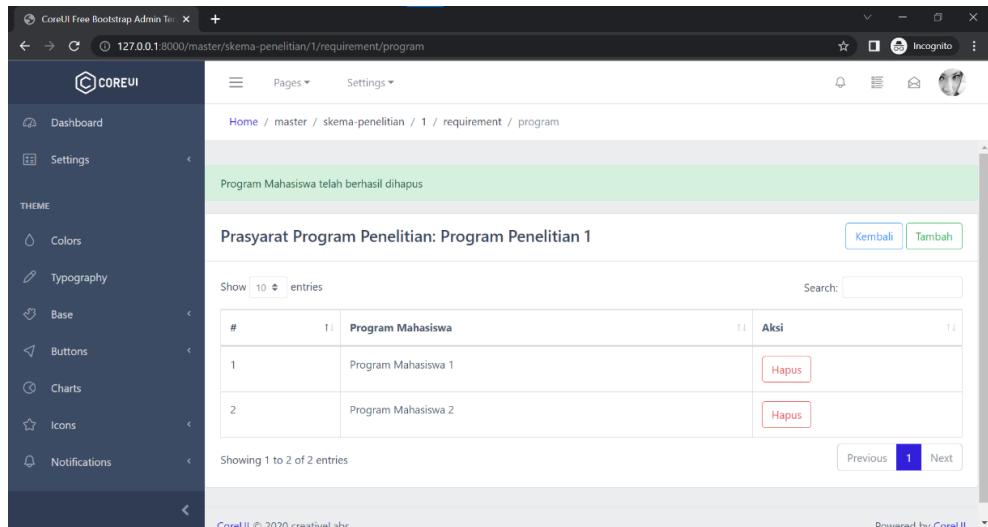
Gambar 4.6. 20 Kode program *controller delete Prasyarat Program Penelitian*

Gambar 4.6.20 adalah kode program yang penulis kerjakan pada *controller* untuk menghapus Program Mahasiswa yang dipilih. Nama fungsinya adalah “store\_research\_req\_sp”. Pada fungsi ini terdapat kode program untuk mencari/find id dari Program Penelitian berdasarkan parameter. Jika id tidak ditemukan, maka akan muncul pesan “Data tidak ditemukan” pada *main page*. Jika id ditemukan, maka Program Mahasiswa tersebut akan dihapus dari prasyarat dengan Teknik *detach*, lalu muncul pesan pada *main page* “Program Mahasiswa telah berhasil dihapus”.



Gambar 4.6. 21 Tampilan SWAL

Gambar 4.6.21 adalah tampilan ketika tombol “Hapus” dari salah satu Program Mahasiswa diklik. Ketika diklik, maka akan memunculkan SWAL (*Sweet Alert*) yang memunculkan pesan konfirmasi terlebih dahulu sebelum Program Mahasiswa tersebut dihapus dari Prasyarat Program Penelitian. Jika sudah yakin untuk menghapus prasyarat tersebut, selanjutnya klik “Hapus” pada SWAL tersebut.



Gambar 4.6.22 Tampilan Program Mahasiswa berhasil dihapus

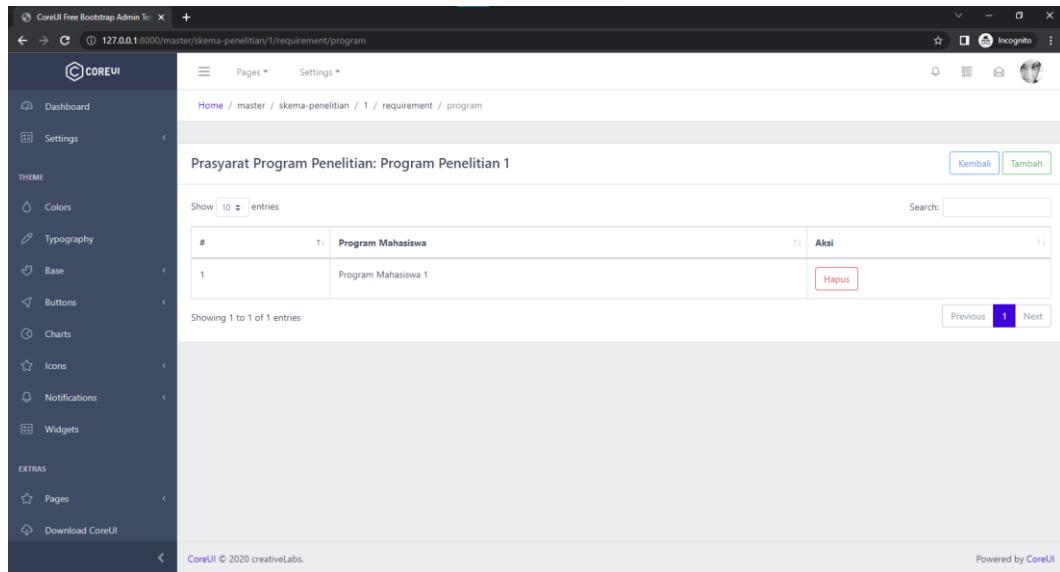
Gambar 4.6.22 adalah tampilan ketika tombol “Hapus” dari SWAL diklik. Ketika diklik, maka Program Mahasiswa tersebut akan terhapus dan menampilkan pesan bahwa Program Mahasiswa tersebut telah berhasil dihapus dari prasyarat.

## 4.7 Pengujian untuk Prasyarat Program Penelitian

Sebelum penulis melakukan *push* ke Github Desktop, penulis tentunya melalukan pengujian terlebih dahulu agar kode program yang penulis kerjakan dapat berjalan dengan baik dan lancar. Jika pada bab sebelumnya telah dibahas mengenai keberhasilan pada saat menambahkan prasyarat, maka pada bab ini penulis akan membahas mengenai pengujian apabila pada saat menambahkan prasyarat terjadi kesalahan.

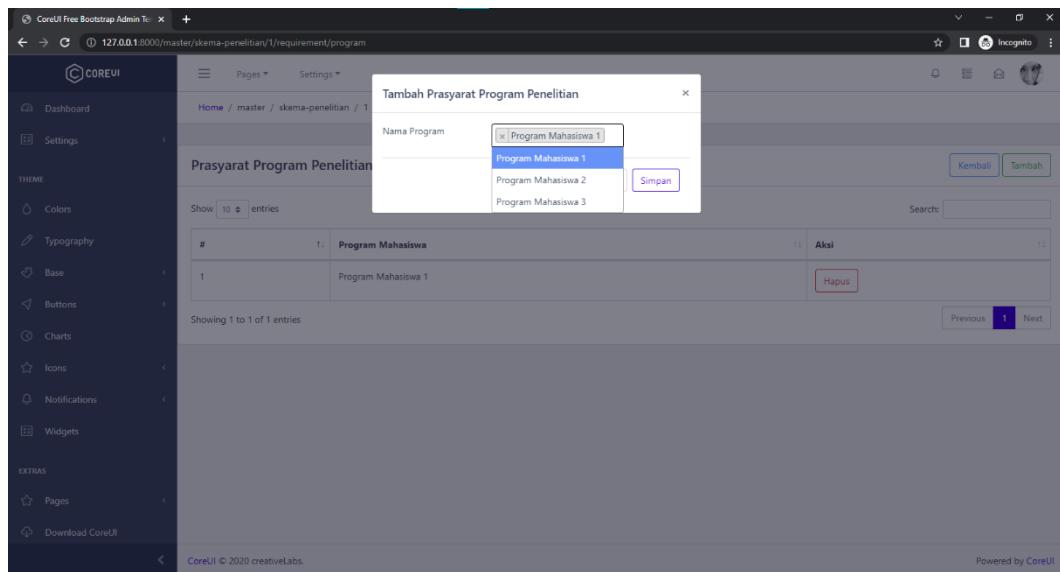
### 4.7.1 Pengujian Menambahkan Data Duplikat

Pertama-tama penulis melakukan pengujian pada saat menambahkan prasyarat Program Penelitian, namun prasyarat yang akan ditambahkan adalah Program Mahasiswa yang sebelumnya sudah ada di *main page*.



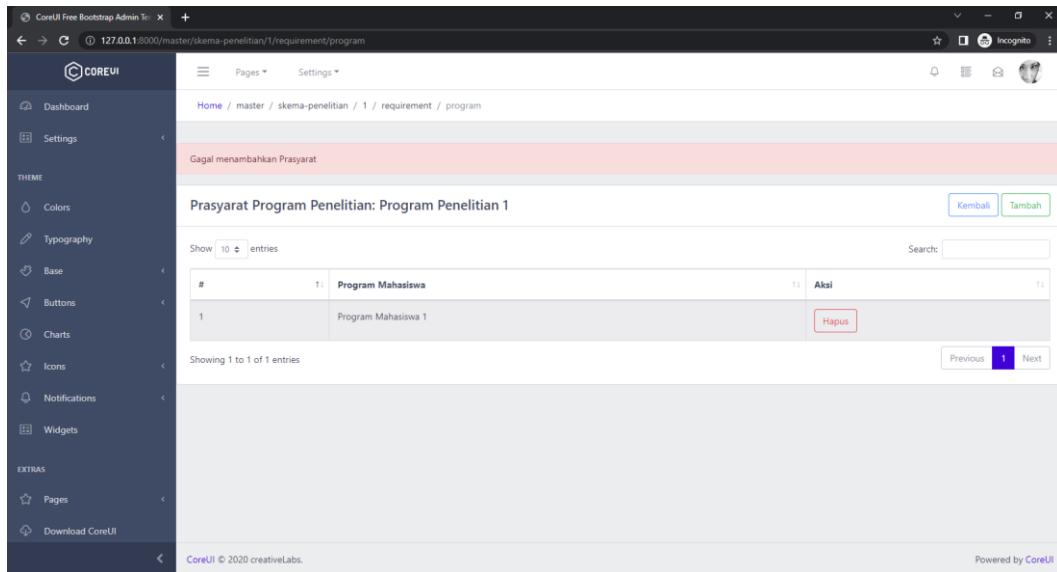
Gambar 4.7. 1 Tampilan sebelum data ditambahkan

Gambar 4.7.1 adalah tampilan awal dari Prasyarat Program Penelitian dari Program Penelitian 1. Program Penelitian 1 baru memiliki 1 prasyarat yaitu Program Mahasiswa. Penulis akan menguji apa yang terjadi apabila penulis menambahkan Program Mahasiswa 1.



Gambar 4.7. 2 Tampilan modal Nama Program duplikat

Gambar 4.7.2 adalah tampilan modal Tambah Prasyarat Program Penelitian ketika ingin menambahkan Program Mahasiswa 1. Lalu penulis akan klik tombol “Simpan”.



Gambar 4.7. 3 Tampilan gagal tambah data duplikat

Gambar 4.7.3 adalah tampilan pada *main page* ketika tombol “Simpan” dari modal diklik. Pada *main page* muncul pesan bahwa prasyarat tersebut gagal untuk ditambahkan. Hal ini dikarenakan kolom `research_scheme_id` dan `student_programme_id` pada tabel `mst_research_req_programme` dibuat menjadi *composite primary key*.

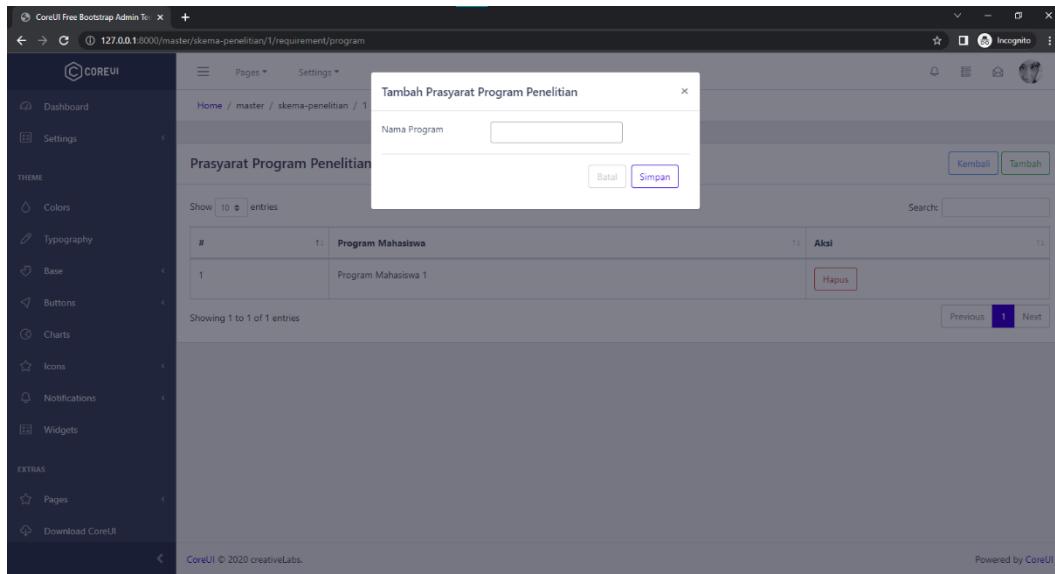
```
lppm=# SELECT * FROM mst_research_req_programmes;
 research_scheme_id | student_programme_id | created_at | updated_at | deleted_at
-----+-----+-----+-----+-----+
 1 | 1 | | | |
(1 row)
```

Gambar 4.7. 4 Tampilan database `mst_research_req_programmes`

Gambar 4.7.4 adalah tampilan pada database untuk tabel `mst_research_req_programmes`. Pada gambar tersebut sudah ada `research_scheme_id` yaitu 1 dan `student_programme_id` yaitu 1.

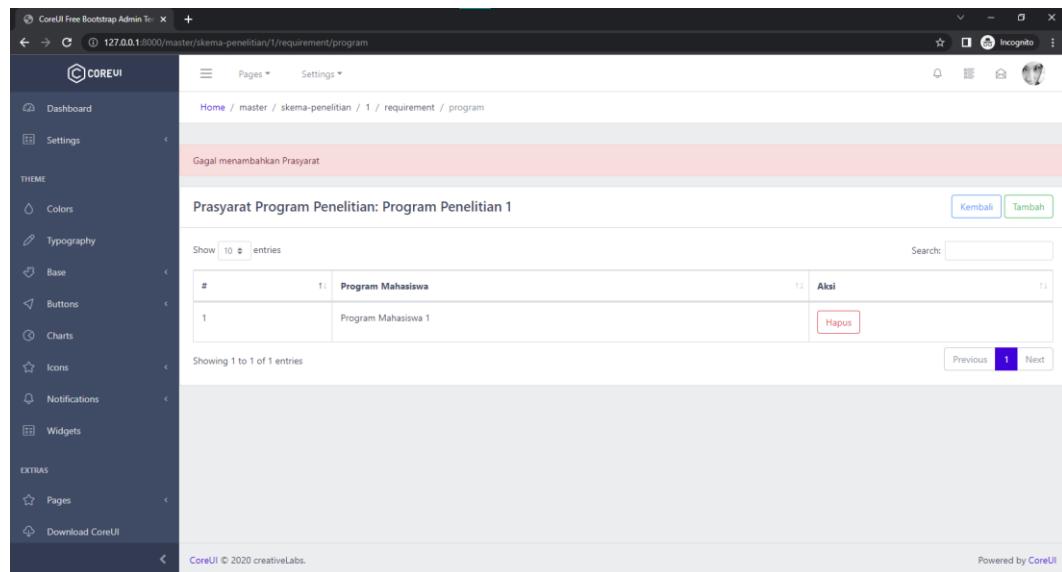
#### 4.7.2 Pengujian Menambahkan Data Kosong

Selanjutnya penulis melakukan pengujian apabila prasyarat yang ditambahkan ke dalam Prasyarat Program Penelitian adalah kosong atau tidak diisi.



Gambar 4.7.5 Tampilan modal Nama Program kosong

Gambar 4.7.5 adalah tampilan ketika ingin menambahkan prasyarat namun Nama Program yang akan dimasukkan tersebut tidak diisi atau kosong. Lalu tombol “Simpan” diklik.



Gambar 4.7.6 Tampilan gagal tambah data kosong

Gambar 4.7.6 adalah tampilan pada *main page* ketika tombol “Simpan” dari modal diklik. Pada *main page* muncul pesan bahwa prasyarat tersebut gagal untuk ditambahkan. Hal ini karena pada *controller* sudah dilakukan antisipasi apabila

Program Mahasiswa dikosongkan maka akan kembali ke *main page* dan memunculkan pesan “Gagal menambahkan prasyarat”.