

MODUL PRAKTIKUM PEMROGRAMAN AKUNTANSI II PROGRAM STUDI SISTEM INFORMASI AKUNTANSI



Fakultas Teknik dan Informatika
Universitas Bina Sarana Informatika
2023

Kata Pengantar

Alhamdulillahirabbil'alamiin, puji syukur kehadiran Tuhan Yang Maha Esa atas segala rahmat dan hidayah-Nya sehingga modul praktikum Pemrograman Akuntansi I ini dapat tersusun hingga selesai. Kami juga mengucapkan terima kasih sebesar-besarnya kepada semua pihak yang telah berkontribusi dalam penyusunan modul ini, baik dalam menyumbangkan pemikiran maupun memberikan motivasi kepada kami untuk dapat menyelesaikan modul ini.

Modul praktikum ini dibuat sebagai pendukung proses pembelajaran matakuliah Pemrograman Akuntansi II bagi mahasiswa/i program studi Sistem Informasi Akuntansi (D3) Fakultas Teknik dan Informatika Universitas Bina Sarana Informatika. Tujuan penyusunan modul ini agar mahasiswa/i merasakan pengalaman dalam pengembangan sistem informasi khususnya sistem informasi akuntansi sehingga mampu mengembangkan kompetensi serta keterampilan dalam pengembangan sistem.

Kami berharap dengan adanya modul praktikum ini dapat menambah pengetahuan dan pengalaman bagi penulis khususnya dan bagi mahasiswa/i pada umumnya. Kami menyadari modul praktikum ini masih banyak sekali kekurangan, oleh karena itu kami sangat memerlukan kritik dan saran atas modul ini untuk pengembangan dan perbaikan modul ini kedepannya.

Akhir kata, kami mengucapkan terima kasih kepada Bapak Dr. Ir.Mochamad Wahyudi, MM, M.Kom, M.Pd, IPU, ASEAN Eng. selaku Rektor Universitas Bina Sarana Informatika beserta jajaran rektorat, Adi Supriyatna, M.Kom selaku ketua program studi Sistem Informasi Akuntansi (D3) dan rekan-rekan dosen Unit Pengembangan Akademik program studi Sistem Informasi Akuntansi yang telah memberikan kesempatan kepada kami dalam penyusunan dan pengembangan modul praktikum ini.

Jakarta, Januari 2023

UNIVERSITAS

Tim Penulis

Daftar Isi

Kata Pengantar	ii
Daftar Isi	iii
PERTEMUAN	1
Setup Awal	1
Instal Tool yang dibutuhkan	1
Perintah - perintah pada Laravel	2
Membuat project baru	3
Install library	5
PERTEMUAN 2	7
Pembuatan database, Model, Tabel	7
Tugas 1	10
Membuat Trigger	13
Pembuatan View Table	14
PERTEMUAN 3	16
Manajemen User (Authentication) dan Templating	16
PERTEMUAN 4	31
Manajemen User (Roles Permission)	31
PERTEMUAN 5	42
Membuat Form Master User	42
PERTEMUAN 6	48
Membuat Form Master Barang	48
PERTEMUAN 7	58
Membuat Form Master Akun	58
Membuat Form Master Setting Akun	68
PERTEMUAN 9	69
Membuat Form Transaksi Pemesanan	69
PERTEMUAN 10 & 11	78
Membuat Form Transaksi Pembelian	78
Membuat Faktur Invoice	86
PERTEMUAN 12	91
Membuat Form Transaksi Retur	91
Membuat Laporan Jurnal Umum Dan Stok Barang	98
PERTEMUAN 13	107
Review pembahasan materi	107
PERTEMUAN 14 & 15	108
Presentasi project	108
Daftar Pustaka	109

PERTEMUAN 1

1.1. Setup Awal Project

A. Tools yang dibutuhkan

Persiapan awal dalam membuat sebuah project pada laravel haruslah tersedia, beberapa syarat yang sudah kita tentukan diantaranya adalah:

1. Xampp Version 7 Up Link(<https://www.apachefriends.org/download.html>)

Catatan: Sesuaikan dengan jenis OS(Windows, Linux dan MacOS)

XAMPP for Windows 7.2.34, 7.3.24 & 7.4.12

Version		Checksum			Size
7.2.34 / PHP 7.2.34	What's Included?	md5	sha1	Download (64 bit)	153 Mb
7.3.24 / PHP 7.3.24	What's Included?	md5	sha1	Download (64 bit)	155 Mb
7.4.12 / PHP 7.4.12	What's Included?	md5	sha1	Download (64 bit)	156 Mb

[Requirements](#) [Add-ons](#) [More Downloads](#) »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

XAMPP for Linux 7.2.34, 7.3.24 & 7.4.12

Version		Checksum			Size
7.2.34 / PHP 7.2.34	What's Included?	md5	sha1	Download (64 bit)	151 Mb
7.3.24 / PHP 7.3.24	What's Included?	md5	sha1	Download (64 bit)	150 Mb
7.4.12 / PHP 7.4.12	What's Included?	md5	sha1	Download (64 bit)	149 Mb

[Requirements](#) [Add-ons](#) [More Downloads](#) »



XAMPP for OS X 7.2.34, 7.3.24, 7.4.12, 7.2.34, 7.3.24 & 7.4.12

Version	Checksum	Size
7.2.34 / PHP 7.2.34	What's Included? md5 sha1	Download (64 bit) 161 Mb
7.3.24 / PHP 7.3.24	What's Included? md5 sha1	Download (64 bit) 161 Mb
7.4.12 / PHP 7.4.12	What's Included? md5 sha1	Download (64 bit) 160 Mb
7.2.34 / PHP 7.2.34	What's Included? md5 sha1	Download (64 bit) 353 Mb
7.3.24 / PHP 7.3.24	What's Included? md5 sha1	Download (64 bit) 356 Mb
7.4.12 / PHP 7.4.12	What's Included? md5 sha1	Download (64 bit) 361 Mb

[Requirements](#) [Add-ons](#) [More Downloads »](#)

2. Install Composer

- Menginstall installer composer 2.0: Download (<https://getcomposer.org/Composer-Setup.exe>)
- Apabila sudah pernah di install silakan di cek terminal command prompt dan ketika perintah **composer** maka akan muncul seperti berikut:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Firmansyah>composer

Composer version 2.2.1 2021-12-22 22:21:31

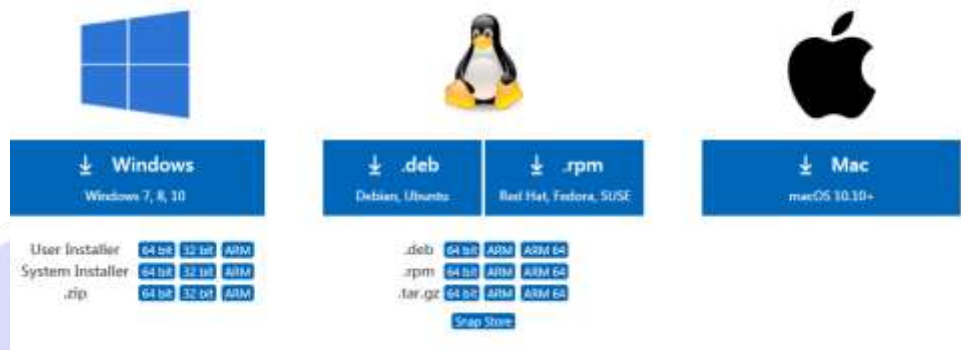
Usage:
  command [optional arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version              Display this application version
      --ansi                Force ANSI output
      --no-ansi             Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
      --profile             Display timing and memory usage information
      --no-plugins           Whether to disable plugins.
      --no-scripts          Skip the execution of all scripts defined in a
```

3. Editor menggunakan Visual Studio Code(VSCode) Link:

<https://code.visualstudio.com/download>

Catatan: Pilih sesuai dengan OS dan versi yang digunakan



4. Install Node.JS

Node.js akan kita gunakan untuk membuat authentication login dan register.

- Windows 7 Link(<https://nodejs.org/download/release/v13.14.0/node-v13.14.0-x64.msi>)

- Windows 10 Link(<https://nodejs.org/en/download/>)

Klik Next dan tunggu hingga selesai instalasi

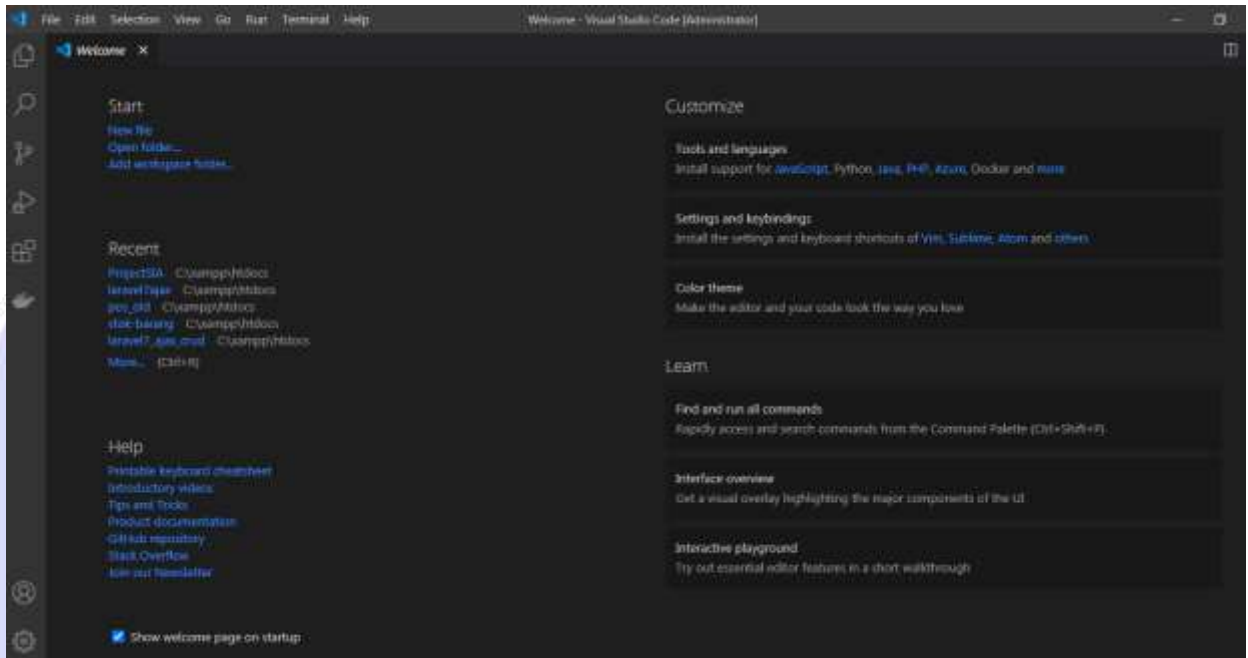


1.2. Perintah – Perintah Umum Pada Laravel

Perintah	Keterangan
php artisan make:model NamaModel	Perintah untuk membuat model
php artisan make:migration NamaTabel	Perintah membuat Tabel dengan migration
php artisan migrate	Perintah untuk menjalankan migrate
php artisan serve	Perintah Untuk Menjalankan Laravel

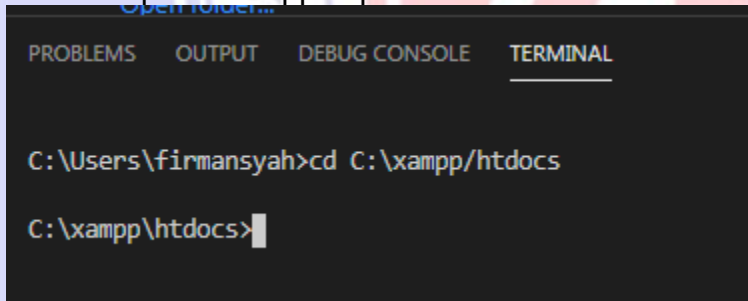
1.3. Membuat Project Baru

1. Dalam pembahasan ini kita sudah siap dengan semua tools yang dibutuhkan, pertama kita buka editor VSCode sebagai berikut:



Tampilan awal VSCode.

- Langkah selanjutnya pilih menu terminal>new terminal kemudian masuk ke folder htdoc pada xampp seperti berikut:



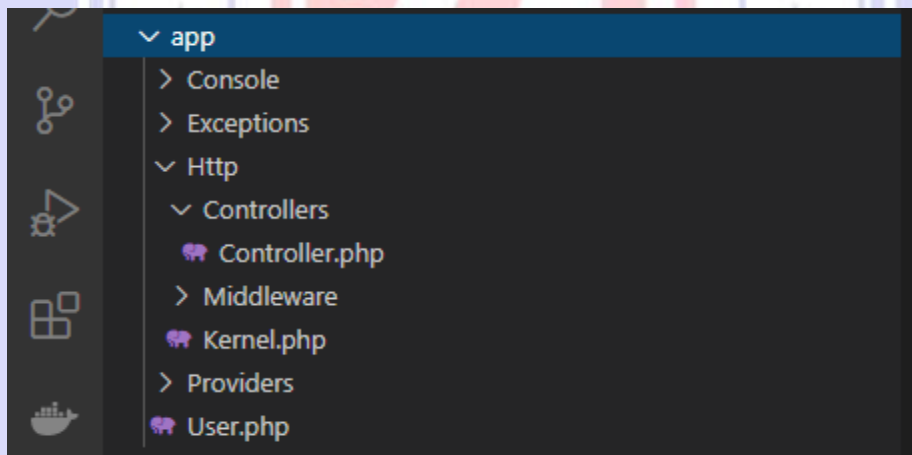
- Kemudian buatlah project baru dengan perintah:

composer create-project --prefer-dist laravel/laravel:^7.0 ProjectSIA2

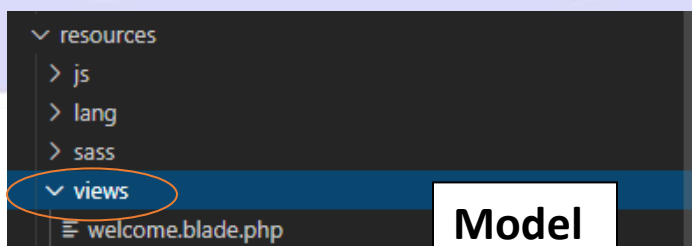
Jika proses install berhasil akan muncul notifikasi pada terminal seperti gambar dibawah ini.

```
Command Prompt
Installing phpunit/php-test-template (1.3.1): Extracting archive
Installing phpunit/php-file-iterator (2.0.3): Extracting archive
Installing thephpunit/tokenizer (1.2.0): Extracting archive
Installing sebastian/code-unit-reverse-lookup (1.0.2): Extracting archive
Installing phpunit/php-code-coverage (7.0.15): Extracting archive
Installing doctrine/instantiator (1.4.0): Extracting archive
Installing phpseclib/phpseclib (1.3.2.2): Extracting archive
Installing phar-io/version (3.0.4): Extracting archive
Installing phar-io/manifest (2.0.3): Extracting archive
Installing symfony/deep-copy (1.10.2): Extracting archive
Installing phpunit/phpunit (8.5.13): Extracting archive
Package suggestions were added by new dependencies, run 'composer suggest' to see details.
Package fransbotto/faker is abandoned, you should avoid using it. No replacement was suggested.
Package phpunit/php-token-stream is abandoned. You should avoid using it. No replacement was suggested.
Generating optimized autoload files
Illuminate\Foundation\ComposerScripts::postAutoloadDump
@php artisan package:discover --ansi
Discovered Packages: facade/ignition
Discovered Packages: fideloper/proxy
Discovered Packages: fruitcake/laravel-cors
Discovered Packages: laravel/tinker
Discovered Packages: nesbot/carbon
Discovered Packages: nunomaduro/collision
Package manifest generated successfully.
Packages you are using are looking for funding.
  in the "composer fund" command to find out more!
@php artisan key:generate --ansi
Application key set successfully.
```

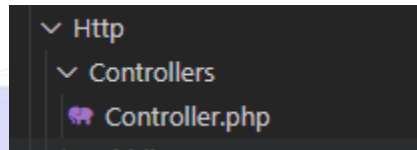
4. Buka folder project tadi pada editor VSCode
Meningat kembali fungsi MVC(Model View Controller) pada Laravel
 - **Model**. Semua file model yang kita buat akan ditempatkan pada folder App.
Fungsi model untuk menangani tabel yang dibuat sebagai suatu class model.



- Khusus untuk manajemen User Interface seperti Layout, dashboard, form dan lainnya yang berhubungan dengan tampilan atau content pada website.



- Selanjutnya adalah Controller, folder ini berada didalam **project/app/http/controller**. Berisi file controller khusus untuk menghandle permintaan dari model seperti Create Read Update dan Delete atau sering kita sebut dengan CRUD.



1.4. Install library

1. Library DOMPDF

Salah satu cara untuk membuat laporan dengan format PDF pada Laravel adalah dengan menggunakan library DOMPDF, library tersebut merupakan library PHP yang bisa digunakan untuk membuat laporan dengan format PDF, konsep DOMPDF adalah mengubah halaman atau konten web menjadi file format PDF. Tahap pertama yang harus dilakukan adalah melakukan pemasangan library DOMPDF pada project dengan cara mendownload package DOMPDF menggunakan composer melalui terminal. Perintah untuk mendownload package DOMPDF adalah sebagai berikut.

Composer require barryvdh/laravel-dompdf

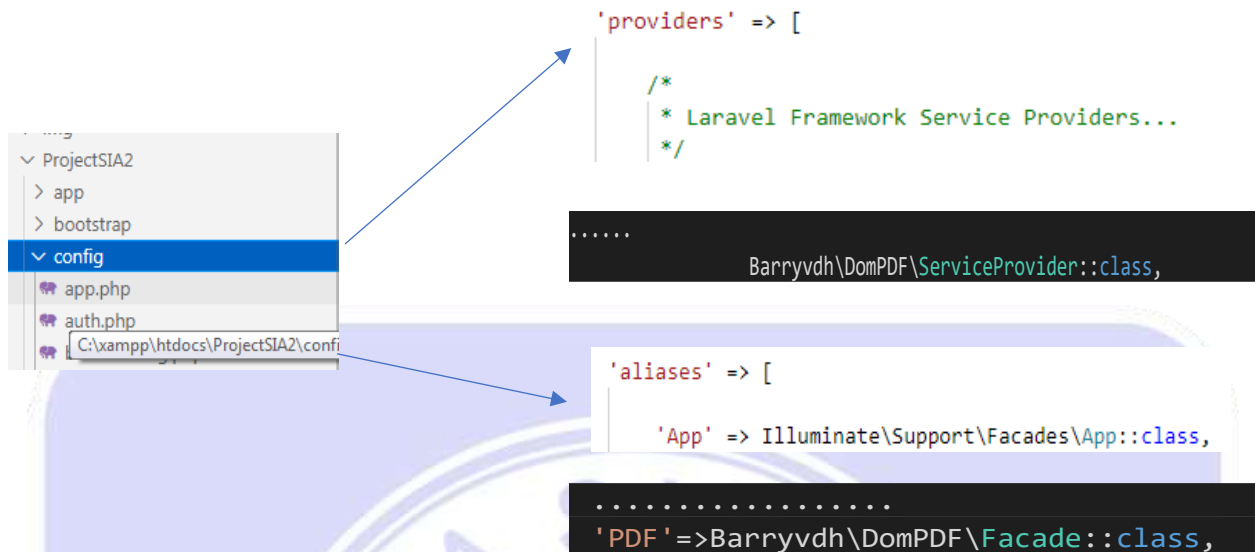
Jika proses download dan pemasangan berhasil akan muncul notifikasi pada terminal seperti gambar dibawah ini.

```
PS C:\xampp\htdocs\ProjectSIAG> composer require barryvdh/laravel-dompdf
Using version ^0.9.0 for barryvdh/laravel-dompdf
./composer.json has been updated
Running composer update barryvdh/laravel-dompdf
Loading composer repositories with package information
Updating dependencies
Lock file operations: 5 installs, 0 updates, 0 removals
  - Locking barryvdh/laravel-dompdf (v0.9.0)
  - Locking dompdf/dompdf (v1.1.1)
  - Locking phenx/php-font-lib (0.5.4)
  - Locking phenx/php-svg-lib (0.5.3)
  - Locking sabberworm/php-css-parser (8.4.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 5 installs, 0 updates, 0 removals
  - Downloading sabberworm/php-css-parser (8.4.0)
  - Downloading phenx/php-svg-lib (0.5.3)
  - Downloading phenx/php-font-lib (0.5.4)
  - Downloading dompdf/dompdf (v1.1.1)
  - Downloading barryvdh/laravel-dompdf (v0.9.0)
 0/5 [-----] 0%
 1/5 [=====] 20%
 2/5 [=====] 40%
 3/5 [=====] 60%
 4/5 [=====] 80%
 5/5 [=====] 100%

- Installing sabberworm/php-css-parser (8.4.0): Extracting archive
- Installing phenx/php-svg-lib (0.5.3): Extracting archive
- Installing phenx/php-font-lib (0.5.4): Extracting archive
- Installing dompdf/dompdf (v1.1.1): Extracting archive
- Installing barryvdh/laravel-dompdf (v0.9.0): Extracting archive
# [>]-----# [>]-----#
2 package suggestions were added by new dependencies, use 'composer suggest' to see details.
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mailer instead.
Package firebase/php-jwt is abandoned, you should avoid using it. No replacement was suggested.
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
+ Illuminate\Foundation\ComposerScripts::postAutoloadDump
+ @php artisan package:discover --ansi
Discovered Package: [2]barryvdh/laravel-dompdf[3]
Discovered Package: [2]facade/ignition[3]
Discovered Package: [2]fidusopen/proxy[3]
Discovered Package: [2]fruitcake/laravel-cors[3]
Discovered Package: [2]laravel/tinker[3]
Discovered Package: [2]mewsdot/carbon[3]
Discovered Package: [2]nunomaduro/collision[3]
[2]Package manifest generated successfully.[3]
60 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
PS C:\xampp\htdocs\ProjectSIAG>
```

Tunggu sampai proses download dan pemasangan package selesai.

Langkah berikutnya adalah lakukan integrasi package DOMPDF dengan project yang dibuat dengan cara menambahkan **class DOMPDF** pada file **app.php** yang terdapat pada folder **"config"**. **Buka file app.php** lalu pada bagian **providers** dan aliases tambahkan kode seperti dibawah ini.



***Tambahkan kode pada baris paling atas/bawah di masing-masing class dan abaikan titik-titik, Kemudian save. Tips agar tidak lupa save klik file>pilih auto save**

```

'providers' => [
    .....
    Barryvdh\DomPDF\ServiceProvider::class,
],
'aliases' => [
    .....
    'PDF' => Barryvdh\DomPDF\Facade::class,
],

```

2. Library Sweet Alert

Library Sweet Alert dapat digunakan untuk menampilkan pesan berhasil atau gagal, agar aplikasi terlihat cantik, berikut perintah untuk install Alert.



Perintah terminal: composer require realrashid/sweet-alert

Jika proses download dan pemasangan berhasil akan muncul notifikasi pada terminal seperti gambar dibawah ini.

```
Package fzaninotto/faker is abandoned, you should avoid using it. No  
Package phpunit/php-token-stream is abandoned, you should avoid using it.  
Generating optimized autoload files  
> Illuminate\Foundation\ComposerScripts::postAutoloadDump  
> @php artisan package:discover --ansi  
Discovered Package: barryvdh/laravel-dompdf  
Discovered Package: facade/ignition  
Discovered Package: fideloper/proxy  
Discovered Package: fruitcake/laravel-cors  
Discovered Package: laravel/tinker  
Discovered Package: nesbot/carbon  
Discovered Package: nunomaduro/collision  
Discovered Package: realrashid/sweet-alert  
Package manifest generated successfully.
```

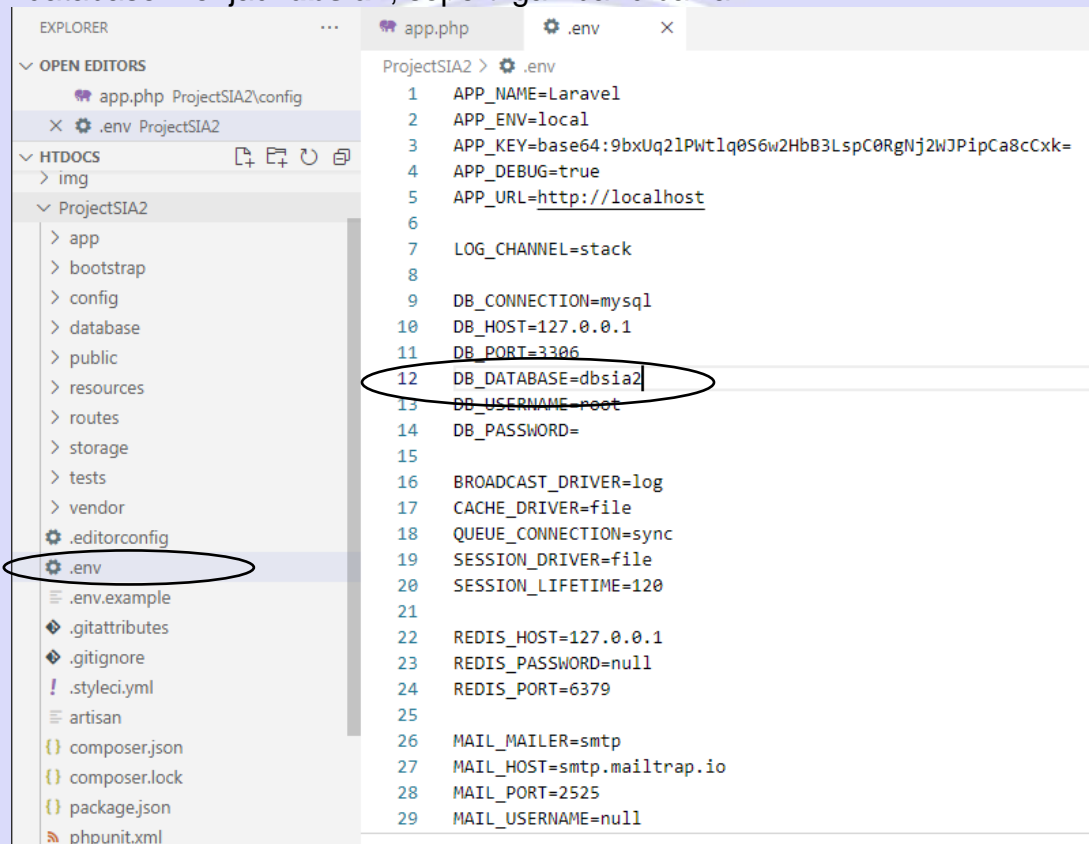


PERTEMUAN 2

2.1. Pembuatan Database, Model dan Tabel

Seperti yang pernah dibahas pada mata kuliah Pemrograman Akuntansi I fungsi dari migration adalah untuk membuat prototype table yang akan kita rancang dan migrasi kedalam database yang kita buat berikut ini adalah contoh merancang dan membuat tabel pada migration:

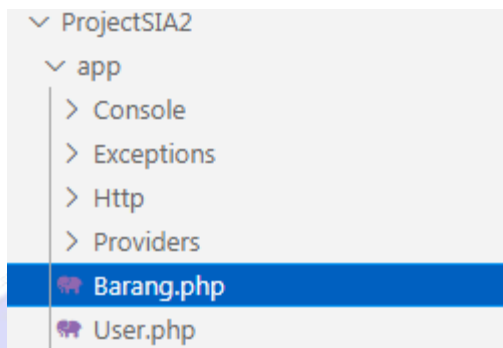
1. Buatlah database baru pada phpmyadmin dengan nama **dbsia2**
2. Lakukan pengaturan koneksi database dengan cara membuka file **.env** lalu rubah database menjadi **dbsia2**, seperti gambar dibawah ini.



3. Selanjutnya buka folder project dengan menggunakan aplikasi visual studio code(VScode) agar lebih memudahkan kita dalam pembuatan script program baik html,Js dan php.
4. Buatlah model beserta file migration sebagai berikut:
Dengan perintah **php artisan make:model NamaModel**
a. Barang

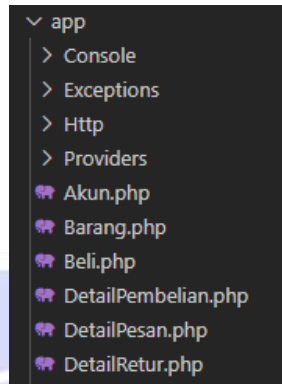
Perintah **php artisan make:model Barang**

```
6.14.4
PS C:\xampp\htdocs\ProjectSIA2> php artisan make:model Barang
Model created successfully.
PS C:\xampp\htdocs\ProjectSIA2>
```



- b. Supplier
Perintah **php artisan make:model Supplier**
- c. Akun
Perintah **php artisan make:model Akun**
- d. Setting
Perintah **php artisan make:model Setting**
- e. Pemesanan
Perintah **php artisan make:model Pemesanan**
- f. DetailPemesanan
Perintah **php artisan make:model DetailPesan**
- g. Pemesanan_tem
Perintah **php artisan make:model Pemesanan_tem**
- h. Temp_pesanan
Perintah **php artisan make:model Temp_pesanan**
- i. Pembelian
Perintah **php artisan make:model Pembelian**
- j. DetailPembelian
Perintah **php artisan make:model DetailPembelian**
- k. Beli
Perintah **php artisan make:model Beli**
- l. Retur
Perintah **php artisan make:model Retur**
- m. DetailRetur
Perintah **php artisan make:model DetailRetur**
- n. Laporan Jurnal
Perintah **php artisan make:model Jurnal**
- o. Laporan
Perintah **php artisan make:model Laporan**

Semua model dapat dilihat pada **app**, seperti pada gambar dibawah ini:



5. Buatlah file migration dengan Skema tabel sebagai berikut:
Dengan perintah: **php artisan make:migration NamaTabel**
Contoh:

php artisan make:migration Barang

apabila sudah dibuat migration barang, maka file dapat dilihat pada direktory:

database/migration/tanggal_pembuatan_migration_Barang.php

Setelah dibuka file migration barang, maka field-field barang dimasukan seperti dibawah ini:

a) Barang

```
public function up()
{
    Schema::create('barang', function (Blueprint $table){
        $table->string('kd_brg',5)->primary();
        $table->string('nm_brg',5);
        $table->integer('harga');
        $table->integer('stok');
    });
}
```

Note:

- Primary adalah kunci utama pada tabel tersebut.
- Type data String yang terdapat dalam migration akan secara otomatis berubah kedalam varchar pada MySQL.

Selanjutnya lakukan migrate dengan perintah **php artisan migrate**. Maka tabel yang kita buat pada scheme migration akan terbentuk pada database **dbisia2**.

TUGAS 1

Buatlah tabel-tabel master dibawah ini dengan contoh diatas:

b) Supplier

No	Field	Type data	Keterangan
1.	kd_supp	varchar(5)	Primary Key
2.	nm_supp	varchar(25)	
3.	alamat	varchar(50)	
4.	telepon	varchar(13)	

c) Akun

No.	Field	Type data	Keterangan
1.	no_akun	varchar(5)	Primary Key
2.	nm_akun	varchar(25)	

d) Setting

No	Field	Type data	Keterangan
1.	Id_setting	varchar(5)	Primary Key
2.	no_akun	varchar(5)	
3.	nama_transaksi	varchar(20)	

e) Pemesanan

No	Field	Type data	Keterangan
1.	no_pesanan	varchar(14)	Primary Key
2.	tgl_pesanan	Date	
3.	Total	Int	
4.	kd_supp	varchar(5)	

f) Detail_pesanan

No	Field	Type data	Keterangan
1.	no_pesanan	varchar(14)	

2.	kd_brg	varchar(5)	
3.	qty_pesanan	Int	
4.	Subtotal	Int	

g) Temp_pemesanan

No	Field	Type data	Keterangan
1.	kd_brg	varchar(5)	
2.	qty_pesanan	Int	

h) Pembelian

No	Field	Type data	Keterangan
1.	no_beli	varchar(14)	Primary Key
2.	tgl_beli	Date	
3.	no_faktur	varchar(14)	
4.	total_beli	Int	
5.	no_pesanan	varchar(14)	

i) detail_pembelian

No	Field	Type data	Keterangan
1.	no_beli	varchar(14)	
2.	kd_brg	varchar(5)	
3.	qty_beli	Int	
4.	sub_beli	Int	

j) retur_beli

No	Field	Type data	Keterangan
1.	no_retur	varchar(14)	Primary Key
2.	tgl_retur	date	
3.	total_retur	Int	

k) detail_retur

No	Field	Type data	Keterangan
1.	no_retur	varchar(14)	
2.	kd_brg	varchar(5)	
3.	qty_retur	Int	
4.	sub_retur	Int	

l) jurnal

No	Field	Type data	Keterangan
1.	no_jurnal	varchar(14)	
2.	keterangan	Text	
3.	tgl_jurnal	Date	
4.	no_akun	varchar(5)	
5.	debit	Int	
6.	kredit	Int	

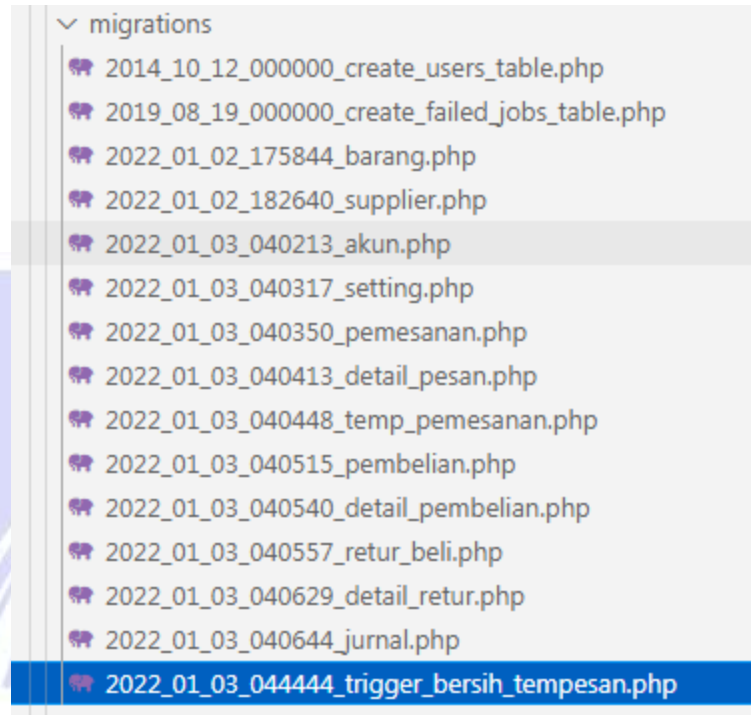
2.2. Membuat Trigger

Trigger adalah merupakan kumpulan sebuah store prosedur yang terdapat dalam mysql bekerja secara otomatis saat user melakukan modifikasi data pada tabel. Modifikasi data yang dilakukan pada tabel yaitu berupa perintah INSERT, UPDATE, dan DELETE. INSERT, UPDATE dan DELETE bisa digabung jadi satu trigger yang dinamakan Multiple Trigger.

Pada project ini, ada 2 Trigger yang akan kita buat yaitu sebagai berikut:

1. Trigger yang pertama kita buat untuk mengosongkan **tabel temp_pembelian**, sama halnya seperti membuat table, dalam pembuatan trigger pun kita akan memanfaatkan fasilitas migration dalam laravel jadi untuk tahapannya relative sama seperti pembuatan table, berikut perintahnya:

php artisan make:migration trigger_bersih_tempesan



Silahkan buka file **trigger_bersih_tempesan**, dan masukan code dibawah ini pada bagian **function up() & function down()**:

```
public function up()
{
    DB::unprepared('
CREATE TRIGGER clear_tem_pesan AFTER INSERT ON detail_pesan
FOR EACH ROW
BEGIN
    DELETE FROM temp_pemesanan;
END
');
}

public function down()
{
    DB::unprepared('DROP TRIGGER clear_tem_pesan');
}
}
```

***Jika kode di copy paste betulkan kembali format teksnya/ketik ulang pada bagian yang disalahkan oleh vsCode**

2. Trigger selanjutnya untuk menambah stok ketika dilakukan pembelian, sama halnya seperti membuat table, dalam pembuatan trigger pun kita akan memanfaatkan

fasilitas migration dalam laravel jadi untuk tahapan nya relative sama seperti pembuatan table, berikut skripnya untuk membuat trigger dalaam penambahan stok

php artisan make:migration trigger_tambah

Silahkan buka file **trigger_tambah**, dan masukan code dibawah ini:

```
public function up()
{
    DB::unprepared('
CREATE TRIGGER update_stok after INSERT ON detail_pembelian
FOR EACH ROW BEGIN
UPDATE barang
SET stok = stok + NEW.qty_beli
WHERE
kd_brg = NEW.kd_brg;
END
');
}

public function down()
{
    DB::unprepared('DROP TRIGGER update_stok');
}
}
```

Selanjutnya lakukan migrate dengan perintah **php artisan migrate**.

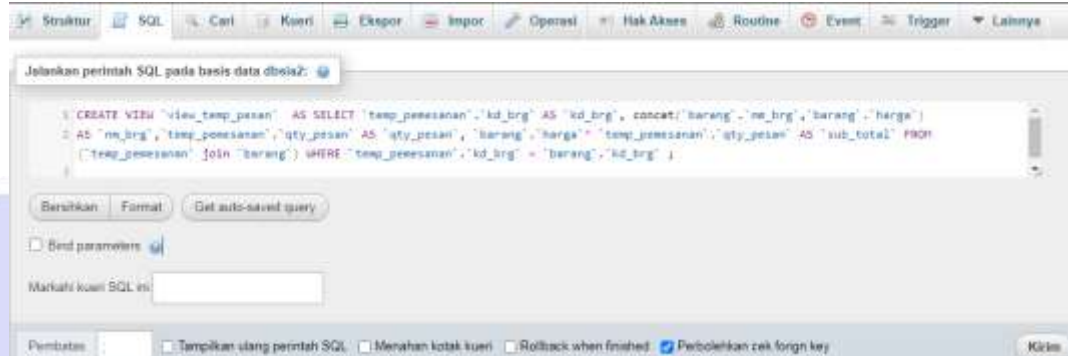
```
PS C:\xampp\htdocs\ProjectSIA2> php artisan migrate
Migrating: 2022_01_03_044444_trigger_bersih_tempesan
Migrated: 2022_01_03_044444_trigger_bersih_tempesan (0.23 seconds)
Migrating: 2022_01_03_045219_trigger_tambah
Migrated: 2022_01_03_045219_trigger_tambah (0.1 seconds)
PS C:\xampp\htdocs\ProjectSIA2> █
```

2.3. Pembuatan View Table

Dalam proyek ini view digunakan untuk menampilkan data yang kita inginkan, langkah membuat View itu terdapat pada **PhpMyadmin**, silahkan buka database yang sudah dibuat yaitu **dbsia2** yang terdapat pada <http://localhost/phpmyadmin/>



Setelah itu, buka database **dbsia2**, dan klik toolbar SQL, maka akan muncul seperti gambar dibawah ini:



Apabila sudah tampil, maka View bisa dibuat, berikut beberapa code View yang akan kita buat dan masukan kedalam editor sql:

a. View Untuk Menampilkan Data Temporary Pemesanan:

```
CREATE VIEW `view_temp_pesan` AS SELECT `temp_pemesanan`.`kd_brg` AS `kd_brg`, concat(`barang`.`nm_brg`,`barang`.`harga`) AS `nm_brg`, `temp_pemesanan`.`qty_pesan` AS `qty_pesan`, `barang`.`harga` * `temp_pemesanan`.`qty_pesan` AS `sub_total` FROM (`temp_pemesanan` join `barang`) WHERE `temp_pemesanan`.`kd_brg` = `barang`.`kd_brg` ;
```

Penjelasan:

CREATE VIEW nama_view = digunakan untuk membuat view

Perintah AS digunakan untuk pemberian alias name

peintah CONCAT digunakan untuk menggabungkan 2 atau lebih fiel menjadi 1 field dengan pengalisan

misal

CONCAT(barang.nm_brg,barang.harga) AS nm_brg

jadi hasil pemanggilan nm_brg adalah namabarang20000

misal

jika kita ingin menambahkan spasi diantara nm_brg dan harga makan kita harus merubah

CONCAT menjadi CONCAT(barang.nm_brg,' ',barang.harga)

jadi hasil pemanggilan nm_brg adalah namabarang 20000

b. View Tampil Data Pemesanan:

```
CREATE VIEW `tampil_pemesanan` AS SELECT `detail_pesan`.`kd_brg` AS `kd_brg`, `detail_pesan`.`no_pesan` AS `no_pesan`, `barang`.`nm_brg` AS `nm_brg`, `detail_pesan`.`qty_pesan` AS `qty_pesan`, `detail_pesan`.`subtotal` AS `sub_total` FROM (`barang` join `detail_pesan`) WHERE `detail_pesan`.`kd_brg` = `barang`.`kd_brg` ;
```

c. View Tampil Pembelian:

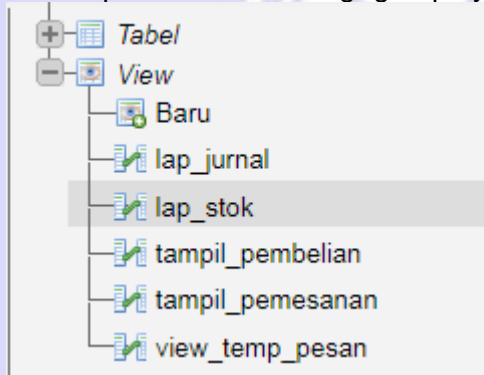
```
CREATE VIEW `tampil_pembelian` AS (select `barang`.`kd_brg` AS `kd_brg`, `detail_pembelian`.`no_beli` AS `no_beli`, `barang`.`nm_brg` AS `nm_brg`, `barang`.`harga` AS `harga`, `detail_pembelian`.`qty_beli` AS `qty_beli` from (`barang` join `detail_pembelian`) where `barang`.`kd_brg` = `detail_pembelian`.`kd_brg` ) ;
```

d. View Laporan Jurnal:

```
CREATE VIEW `lap_jurnal` AS SELECT `akun`.`nm_akun` AS `nm_akun`,  
`jurnal`.`tgl_jurnal` AS `tgl`, sum(`jurnal`.`debit`) AS `debit`, sum(`jurnal`.`kredit`) AS  
`kredit` FROM (`akun` join `jurnal`) WHERE `akun`.`no_akun` = `jurnal`.`no_akun`  
GROUP BY `jurnal`.`no_jurnal` ;
```

e. View Laporan Stok:

```
CREATE VIEW `lap_stok` AS (select `barang`.`kd_brg` AS `kd_brg`,`barang`.`nm_brg`  
AS `nm_brg`,`barang`.`harga` AS `harga`,`barang`.`stok` AS  
`stok`,sum(`detail_pembelian`.`qty beli`) AS `beli`,sum(`detail_retur`.`qty_retur`) AS  
`retur` from ((`barang` join `detail_retur`) join `detail_pembelian`) where  
`barang`.`kd_brg` = `detail_retur`.`kd_brg` and `barang`.`kd_brg` =  
`detail_pembelian`.`kd_brg` group by `barang`.`kd_brg`);
```



PERTEMUAN 3

3.1. Manajemen User(Authentication)

A. Membuat Form login dan Register

Seperti pada pemrograman akuntansi 1 kita sudah pernah mempelajari bagaimana membuat sebuah halaman login dan register, kali ini kita akan mencoba membangun dari awal dengan menggunakan tools seperti ui dan auth. Pada dasarnya laravel sudah menyediakan beberapa tools yang bisa dikombinasikan dalam membuat sebuah manajemen user seperti vue dan npm, berikut langkahnya:

1) Menginstall Authentication

Membuat Ui(User Interface) adalah tahapan awal kita dalam membangun sebuah program berbasis website, fungsi Auth pada laravel yaitu untuk manajemen user/pengguna seperti admin,staff/kasir yang masing-masing memiliki hak aksesnya, berikut tahapannya:

composer require laravel/ui:^2.4

Hasil kompilasi perintah diatas akan melakukan pemasangan laravel/ui versi 2.4, jika berhasil maka akan tampil seperti gambar dibawah ini:

```
PS C:\xampp\htdocs\ProjectSIA2> composer require laravel/ui:^2.4
./composer.json has been updated
Running composer update laravel/ui
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/ui (v2.4.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/ui (v2.4.0)
- Installing laravel/ui (v2.4.0): Extracting archive
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mailer instead.
Package franinotto/faker is abandoned, you should avoid using it. No replacement was suggested.
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: [32mbarryvdh/laravel-dompdf[39m
Discovered Package: [32mfacade/ignition[39m
Discovered Package: [32mfideloper/proxy[39m
Discovered Package: [32mfruitcake/laravel-cors[39m
Discovered Package: [32mlaravel/tinker[39m
Discovered Package: [32mlaravel/ui[39m
Discovered Package: [32mesbot/carbon[39m
Discovered Package: [32mnunomaduro/collision[39m
Discovered Package: [32mrealrashid/sweet-alert[39m
[32mPackage manifest generated successfully.[39m
70 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
PS C:\xampp\htdocs\ProjectSIA2>
```

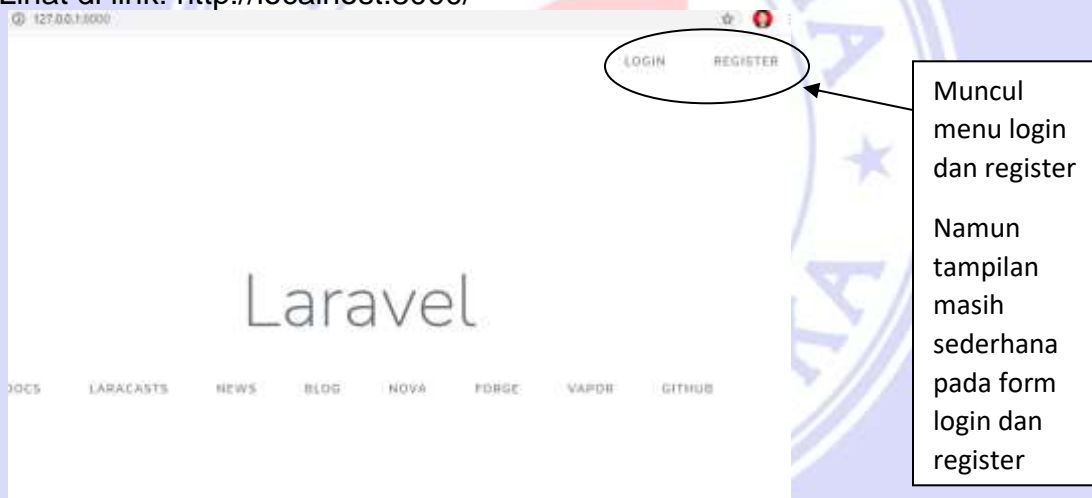
Langkah selanjutnya adalah membuat fitur authentication dengan perintah dibawah ini.

php artisan ui vue --auth

Perintah diatas digunakan untuk membuat beberapa file yang digunakan untuk user authentication seperti form login, form register dan lupa password. Jika perintah tersebut berhasil dijalankan maka akan tampil notifikasi dan membuat beberapa file seperti dibawah ini:

```
PS C:\xampp\htdocs\ProjectSIA2> php artisan ui vue --auth
Vue scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.
PS C:\xampp\htdocs\ProjectSIA2> php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Mon Jan 3 12:31:10 2022] 127.0.0.1:61000 [200]: /favicon.ico
```

Silahkan berikan perintah pada terminal: **php artisan serve** untuk melihat hasilnya
Hasilnya adalah sebagai berikut:
Lihat di link: <http://localhost:8000/>



The image shows two side-by-side screenshots of a Laravel application's login and register pages. The browser's address bar at the top indicates the URL is '127.0.0.1:8000/login'. The left screenshot shows the login page with fields for 'E-Mail Address' and 'Password', a 'Remember Me' checkbox, and a 'Login' button. The right screenshot shows the register page with fields for 'Name', 'E-Mail Address', 'Password', 'Confirm Password', and a 'Register' button.

Tampilan login dan register masih sederhana

2) Membuat Laman Login, register

Selanjutnya kita install modul npm untuk menambah tampilan dari bootstrap dan Js, namun sbelum install kita harus memastikan terlebih dahulu apa sudah terinstall atau belum tools-tools yang diperlukan, berikut perintahnya:

```
PS C:\xampp\htdocs\ProjekLaravel> node -v
v14.15.4
PS C:\xampp\htdocs\ProjekLaravel> npm -v
6.14.10
PS C:\xampp\htdocs\ProjekLaravel> composer
```

Perintah	Fungsi
node -v	Untuk mengecek Node JS
npm -v	Untuk mengecek NPM
composer	Untuk mengecek Composer

Jika muncul semua tools yang diperlukan seperti gambar di atas, maka proses instalasi Node Js, Npm dan composer sudah berhasil.

Langkah berikutnya Integrasi Vue Js di Laravel 7 ketikkan perintah dibawah pada terminal:

Perintah terminal: php artisan ui vue

```
PS C:\xampp\htdocs\ProjekLaravel> php artisan ui vue
Vue scaffolding installed successfully.
```

Apabila sudah sukses mengintegrasikan Vue Js, maka langkah berikutnya proses Install Node Js, berikut perintanya:

Perintah terminal: npm install

```
PS C:\xampp\htdocs\latihanPA2> npm install
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated popper.js@1.16.1: You can find the new Popper v2 at @popperjs/core, this package is dedicated to the legacy v1
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+, Upgrade to chokidar 3 with 15x less dependencies.
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to fsevents 2.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.7 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\webpack\node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
```

setelah selesai lalu ketik perintah dibawah ini untuk compiling dan testing pada proyek, berikut perintahnya:

Perintah terminal: npm run dev

Maka akan muncul seperti gambar dibawah ini:

```
PS C:\xampp\htdocs\latihanPA2> npm run dev

> @ dev C:\xampp\htdocs\latihanPA2
> npm run development

> @ development C:\xampp\htdocs\latihanPA2
> cross-env NODE_ENV=development node_modules\webpack\bin\webpack.js --progress --hide-modules
--config=node_modules/laravel-mix/setup/webpack.config.js

98% after emitting SizeLimitsPlugin
[Done] Compiled successfully in 12166ms 11:32:00

Asset      Size  Chunks  Chunk Names
/css/app.css 170 KIB  /js/app [emitted]  /js/app
t:\Windows\Start Menu\Programs\SnoreToast\0.7.0\SnoreToast.lnk "C:\
\xampp\htdocs\latihanPA2\node_modules\node-notifier\vendor\snoretoast\Windows\Start Men
st\snoretoast-x64.exe" Snore.ToastToasts.0.7.0 anPA2\node_modules\
PS C:\xampp\htdocs\latihanPA2> php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Tue Dec 15 11:32:50 2020] PHP 7.4.13 Development Server (http://127.0.0.1:8000) started
```

Gambar Proses Install modul npm dan run npm. Setelah proses instal selesai, lalu ketikan perintah dibawah ini:

Perintah terminal: Php artisan migrate

Apabila perintah diatas berhasil maka akan tampil seperti gambar dibawah ini:

Asset	Size	Chunks	Chunk Names
/css/app.css	180 KiB	/js/app [emitted]	/js/app
/js/app.js	1.4 MiB	/js/app [emitted]	/js/app

```

PS C:\xampp\htdocs\ProjectSIA2> php artisan migrate
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (1.86 seconds)
PS C:\xampp\htdocs\ProjectSIA2>

```

Selanjutnya lihat hasilnya ketikkan perintah: `php artisan serve` dan buka browser di <http://127.0.0.1:8000/> Atau localhost:8000

Login

E-Mail Address

Password

☐ Remember Me

Login

[Forgot Your Password?](#)

Laman login pada home

Register

Name

E-Mail Address

Password

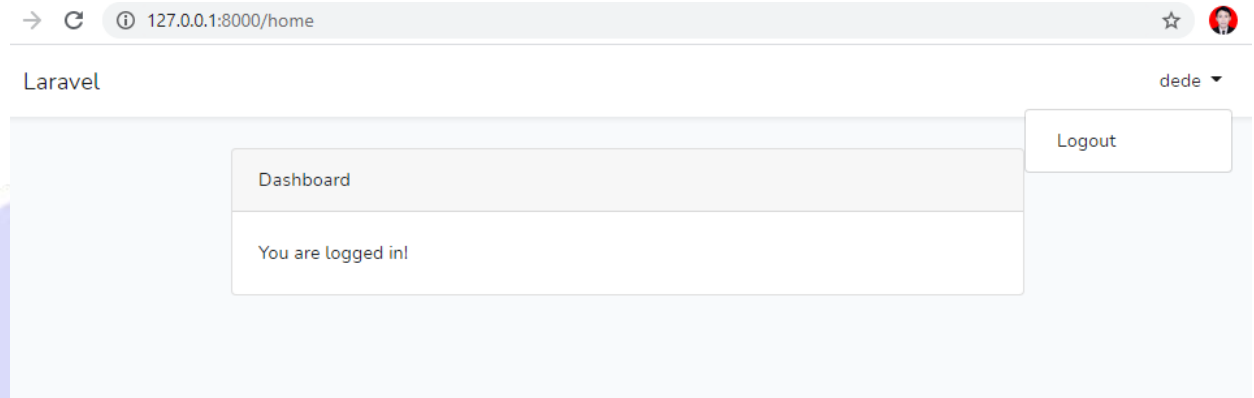
Confirm Password

Register

Laman register pada home

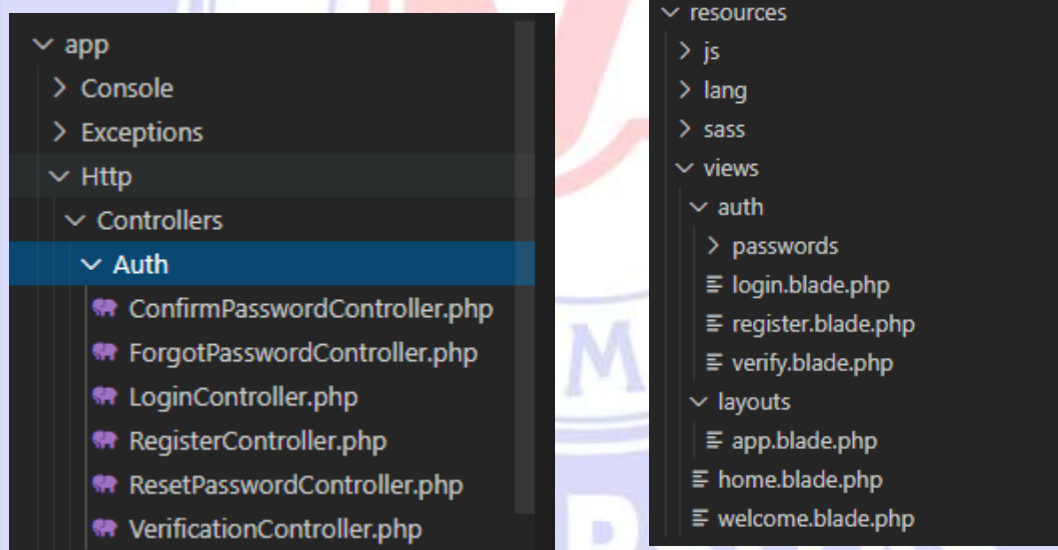
Selanjutnya silakan coba lakukan register user dan cobalah untuk login kembali dengan user yang sudah di registrasikan sebagai berikut:

Maka akan tampil sebagai berikut:



Laman dashboard setelah login. Mudah bukan? Salah satu kelebihan Laravel dapat dependency dengan modul lainnya seperti vue.js dan node.js juga untuk akses login kita sudah disediakan fungsi **Auth(Authentication)** sehingga memudahkan kita dalam mengelola user.

Resource yang terbentuk diantaranya adalah:



File controller login dan register serta views File login, register dan layout

3) Templating dan Manajemen User

Dalam pembuatan templating seperti pada pembahasan pada matakuliah Pemrograman Akuntansi 1 di semester lalu kita sudah dapat template yang bisa kita gunakan untuk project kita kali ini, silakan unduh di link berikut:

Nama ↑

asset

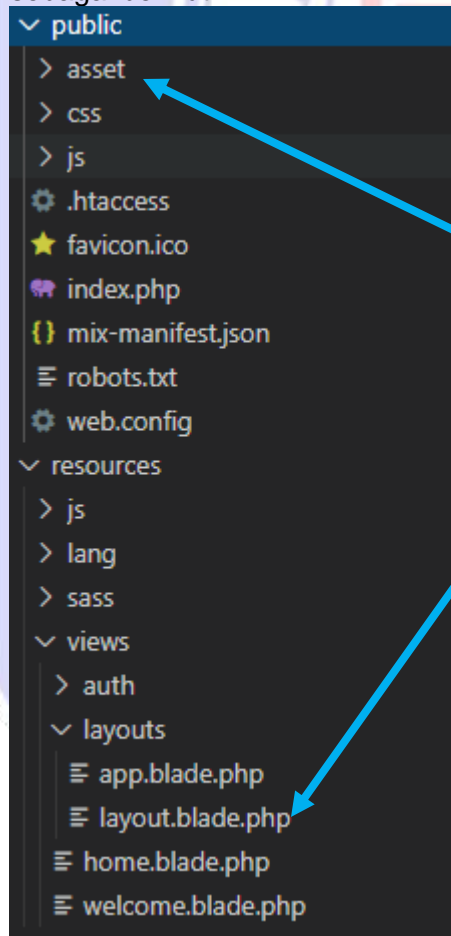
layout.blade.php

Link : <https://bit.ly/TemplatePA2>

Folder asset berisi kumpulan library yang digunakan untuk mendukung tampilan pada template yang berisi file CSS, JS, JQuery, gambar dan sebagainya. Sedangkan file **layout.blade.php** merupakan file yang berisi kode untuk menampilkan halaman dashboard.

Copy folder asset yang terdapat pada template lalu paste ke dalam folder **C:\xampp\htdocs\ProjectSIA2\public**. Kemudian copy file **layout.blade.php** pada folder template lalu paste pada folder layouts:

C:\xampp\htdocs\ProjectSIA2\resources\views\layouts. Sehingga struktur folder sebagai berikut:



Nama ↑

asset

layout.blade.php

Halaman form login sudah tersedia hasil kompilasi perintah **php artisan ui vue --auth**, namun tampilan belum disesuaikan dengan template yang ada. Untuk menyesuaikan tampilan halaman login dengan templatanya, buka file **login.blade.php** yang berada pada folder “resources/views/auth” lalu ganti kode yang ada dengan kode berikut ini.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">
  <title>Pemrograman Akuntansi II</title>
  <!-- Custom fonts for this template-->
  <link href="{{ asset('asset/vendor/fontawesome-free/css/all.min.css')}}" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">
  <!-- Custom styles for this template-->
  <link href="{{ asset('asset/css/sb-admin-2.min.css')}}" rel="stylesheet">
</head>
<body class="bg-gradient-dark">
  <div class="container">
    <!-- Outer Row -->
    <div class="row justify-content-center">
      <div class="col-xl-5 col-lg-12 col-md-9">
        <div class="card o-hidden border-0 shadow-lg my-5">
          <div class="card-body p-0">
            <!-- Nested Row within Card Body -->
            <div class="center">
              <div class="col-lg-6 d-none d-lg-block "></div>
              <div class="col-lg-20">
                <div class="p-5">
                  <div class="text-center">
                    <h1 class="h4 text-gray-900 mb-4">Sistem Informasi Akuntansi<br>Fakultas Teknik dan Informatika<br>
                    <br></h1>
                  </div>
                  <form method="POST" action="{{ route('login') }}">
                    @csrf
                    <div class="form-group row">
                      <label for="email" class="col-md-12 col-form-label text-md-left">{{ __('E-Mail Address') }}</label>
```

```

        <div class="col-md-12">
<input id="email" type="email" class="form-control @error('email') is-
invalid @enderror" name="email" value="{{ old('email') }}" required autocomplete="email" autofocus>
        @error('email')
            <span class="invalid-feedback" role="alert">
                <strong>{{ $message }}</strong>
            </span>
        @enderror
    </div>
</div>
<div class="form-group row">
    <label for="password" class="col-md-12 col-form-label text-md-
left">{{ __('Password') }}</label>
    <div class="col-md-12">
        <input id="password" type="password" class="form-
control @error('password') is-invalid @enderror" name="password" required autocomplete="current-
password">
        @error('password')
            <span class="invalid-feedback" role="alert">
                <strong>{{ $message }}</strong>
            </span>
        @enderror
    </div>
</div>
<div class="form-group row">
    <div class="col-md-12 offset-md-12">
        <div class="form-check">
            <input class="form-check-
input" type="checkbox" name="remember" id="remember" {{ old('remember') ? 'checked' : '' }}>

            <label class="form-check-label" for="remember">
                {{ __('Remember Me') }}
            </label>
        </div>
    </div>
</div>
<div class="form-group row mb-0">
    <div class="col-md-12 offset-md-12">
        <button type="submit" class="btn btn-primary">
            {{ __('Login') }}
        </button>
        @if (Route::has('password.request'))
            <a class="btn btn-link" href="{{ route('password.request') }}">
                {{ __('Forgot Your Password?') }}
            </a>
        @endif
    </div>
</div>

```

```

        </a>
    @endif
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- Bootstrap core JavaScript-->
<script src="{{ asset('asset/vendor/jquery/jquery.min.js')}}"></script>
<script src="{{ asset('asset/vendor/bootstrap/js/bootstrap.bundle.min.js')}}"></script>
<!-- Core plugin JavaScript-->
<script src="{{ asset('asset/vendor/jquery-easing/jquery.easing.min.js')}}"></script>
<!-- Custom scripts for all pages-->
<script src="{{ asset('asset/js/sb-admin-2.min.js')}}"></script>

</body>
</html>

```

Untuk melihat hasilnya klik login pada halaman depan Laravel, atau masukan url 127.0.0.1:8000/login pada web browser, maka tampilan halaman login yang baru menjadi seperti gambar dibawah ini. jangan lupa jalankan perintah command terminal dengan perintah **php artisan serve**

UNIVERSITAS

Sistem Informasi Akuntansi
Fakultas Teknik dan Informatika



E-Mail Address

Password

☐ Remember Me

[Forgot Your Password?](#)

Kemudian kita lanjutkan pada laman register yang belum kita rubah, buka file blade register pada folder **projectSIA2/Auth/register.blade.php** dan rubah kodenya sebagai berikut:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">
  <title>Pemrograman Akuntansi II</title>
  <!-- Custom fonts for this template-->
  <link href="{{ asset('asset/vendor/fontawesome-free/css/all.min.css')}}" rel="stylesheet" type="text/css">
  <link
    href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i"
    rel="stylesheet">
  <!-- Custom styles for this template-->
  <link href="{{ asset('asset/css/sb-admin-2.min.css')}}" rel="stylesheet">
</head>
```

```

<body class="bg-gradient-dark">
  <div class="container">
    <!-- Outer Row -->
    <div class="row justify-content-center">
      <div class="col-xl-5 col-lg-12 col-md-9">
        <div class="card o-hidden border-0 shadow-lg my-5">
          <div class="card-body p-0">
            <!-- Nested Row within Card Body -->
            <div class="center">
              <div class="col-lg-6 d-none d-lg-block "></div>
              <div class="col-lg-20">
                <div class="p-5">
                  <div class="text-center">
                    <h1 class="h4 text-gray-900 mb-4">Sistem Informasi Akuntansi<br>Fakultas Teknik
                    dan Informatika<br>
                    <br></h1>
                  </div>
                  <form method="POST" action="{{ route('register') }}">
                    @csrf

                    <div class="form-group row">
                      <label for="name"
                        class="col-md-4 col-form-label text-md-right">{{ __('Name') }}</label>

                      <div class="col-md-6">
                        <input id="name" type="text"
                          class="form-control @error('name') is-invalid @enderror" name="name"
                          value="{{ old('name') }}" required autocomplete="name" autofocus>

                        @error('name')
                        <span class="invalid-feedback" role="alert">
                          <strong>{{ $message }}</strong>
                        </span>
                        @enderror
                      </div>
                    </div>

                    <div class="form-group row">
                      <label for="email"
                        class="col-md-4 col-form-label text-md-right">{{ __('E-
Mail Address') }}</label>

                      <div class="col-md-6">

```



```

<input id="email" type="email"
      class="form-control @error('email') is-invalid @enderror"
      name="email" value="{{ old('email') }}" required
      autocomplete="email">

      @error('email')
      <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
      </span>
      @enderror
    </div>
  </div>

  <div class="form-group row">
    <label for="password"
      class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>

    <div class="col-md-6">
      <input id="password" type="password"
        class="form-control @error('password') is-invalid @enderror"
        name="password" required autocomplete="new-password">

      @error('password')
      <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
      </span>
      @enderror
    </div>
  </div>

  <div class="form-group row">
    <label for="password-confirm"
      class="col-md-4 col-form-label text-md-
right">{{ __('Confirm Password') }}</label>

    <div class="col-md-6">
      <input id="password-confirm" type="password" class="form-control"
        name="password_confirmation" required autocomplete="new-
password">

      </div>
    </div>

    <div class="form-group row mb-0">
      <div class="col-md-6 offset-md-4">

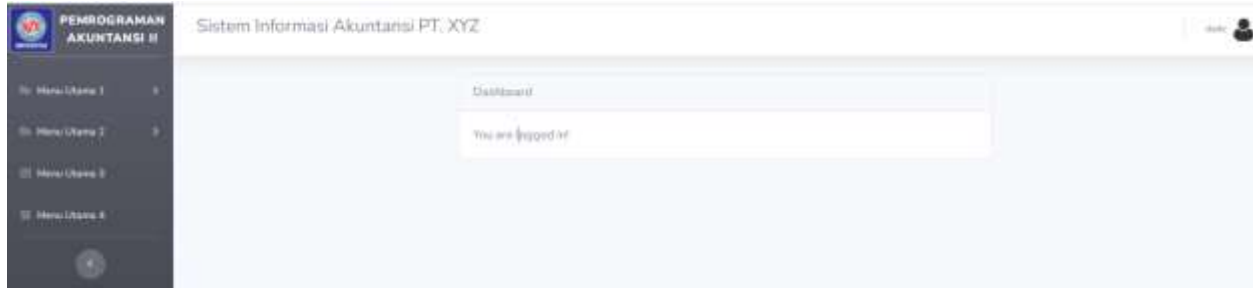
```

Lalu lakukan refresh laman web kembali untuk melihat hasil, seperti berikut:

Hasil tampilan register yang dirubah. Tahapan selanjutnya, rubahlah kode pada file [home.blade.php](#) pada baris:

`@extends('layouts.app')` menjadi `extends('layouts.layout')`

Kemudian coba lakukan register akun dan lakukan login kembali ke laman login untuk memastikan data user ter Autentikasi dengan benar. Jika benar maka akan dapat melakukan login seperti pada tampilan dashboard sebagai berikut:



PERTEMUAN 4

4.1 Manajemen User(Roles Permission)

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Using version ^5.4 for spatie/laravel-permission
./composer.json has been updated
Running composer update spatie/laravel-permission
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking spatie/laravel-permission (5.4.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading spatie/laravel-permission (5.4.2)
  - Installing spatie/laravel-permission (5.4.2): Extracting archive
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mailer instead.
Package fzaninotto/faker is abandoned, you should avoid using it. No replacement was suggested.
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: [32mbarryvdh/laravel-dompdf[39m
Discovered Package: [32mfacade/ignition[39m
Discovered Package: [32mfideloper/proxy[39m
Discovered Package: [32mfruitcake/laravel-cors[39m
Discovered Package: [32mlaravel/tinker[39m
Discovered Package: [32mlaravel/ui[39m
Discovered Package: [32mnnesbot/carbon[39m
Discovered Package: [32mnunomaduro/collision[39m
Discovered Package: [32mrealrashid/sweet-alert[39m
Discovered Package: [32mspatie/laravel-permission[39m
[32mPackage manifest generated successfully.[39m
71 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
PS C:\xampp\htdocs\ProjectSIA2> []
```

Permission

1. Role

adalah sebuah istilah dalam control akses ketika kita mengunjungi sebuah website, dimana setiap user(Pengguna) memiliki level akses yang berbeza, maaf berbeda maksudnya. Seperti contoh dalam sebuah situs belanja online kita disuguhkan menu katalog barang, keranjang belanja, whistlist(yang ga dibeli-beli) konfirmasi pembayaran hingga dengan review barang. Sedangkan berbeda dengan sisi user sebagai pedagang seperti pemberitahuan pembelian, konfirmasi pembelian sampai dengan pengiriman barang. Nah itulah beberapa akses yang dikontrol dalam manajemen user pada sebuah website sehingga dapat kita akses dengan mudah dan dinamis

menurut dengan kebutuhan kita masing-masing, baiklah tidak berlama-lama lagi langsung saja kita buat contoh kasus dalam membuat permission di project pemrograman akuntansi 2 ini, berikut adalah langkah-langkahnya cekidot:

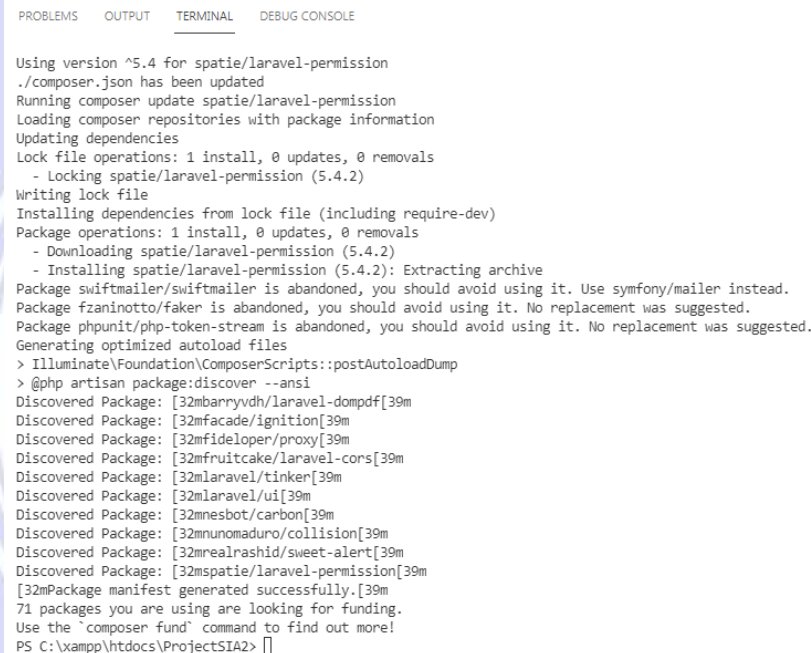
2. Modul Role Permission

Membuat role permission pada Laravel:

1) Kita download modul via Laravel command dengan perintah:

Perintah terminal: `composer require spatie/laravel-permission`

Apabila proses telah selesai, maka akan tampil seperti gambar di bawah ini:



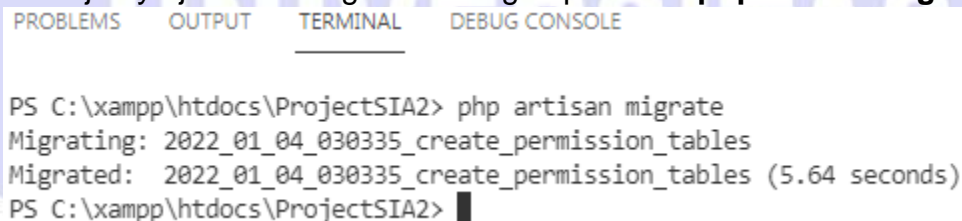
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Using version ^5.4 for spatie/laravel-permission
./composer.json has been updated
Running composer update spatie/laravel-permission
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking spatie/laravel-permission (5.4.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading spatie/laravel-permission (5.4.2)
- Installing spatie/laravel-permission (5.4.2): Extracting archive
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mailer instead.
Package fzaninotto/faker is abandoned, you should avoid using it. No replacement was suggested.
Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: [32mbarryvdh/laravel-dompdf[39m
Discovered Package: [32mfacade/ignition[39m
Discovered Package: [32mfideloper/proxy[39m
Discovered Package: [32mfruitcake/laravel-cors[39m
Discovered Package: [32mlaravel/tinker[39m
Discovered Package: [32mlaravel/ui[39m
Discovered Package: [32mmesbot/carbon[39m
Discovered Package: [32minunomaduro/collision[39m
Discovered Package: [32mrealrashid/sweet-alert[39m
Discovered Package: [32mspatie/laravel-permission[39m
[32mPackage manifest generated successfully.[39m
71 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
PS C:\xampp\htdocs\ProjectSIA2>
```

2) Selanjutnya kita publish/register file migrationnya dari modul ini dengan perintah:

**php artisan vendor:publish --
provider="Spatie\Permission\PermissionServiceProvider"**

Selanjutnya jalankan migration dengan perintah: **php artisan migrate**



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\xampp\htdocs\ProjectSIA2> php artisan migrate
Migrating: 2022_01_04_030335_create_permission_tables
Migrated: 2022_01_04_030335_create_permission_tables (5.64 seconds)
PS C:\xampp\htdocs\ProjectSIA2>
```

Dan apabila provider spatie tidak masuk kedalam library package maka kita masukan manual. Untuk memastikan ada atau tidak silakan dicek dibagian folder **config/app.php file:** dan masukan kode berikut pada bagian atas/bawah:

`Spatie\Permission\PermissionServiceProvider::class,`

3) Penggunaan

Setelah package spatie laravel permission dipastikan sudah ada pada config maka selanjutnya Untuk menggunakan package Laravel Permission ini kita tambahkan dulu **trait HasRoles** ke model **User**(Model ini berfungsi merelasikan antara user dengan roles). Buka file model User dan tambahkan kode dibawah:

```
<?php

namespace App;

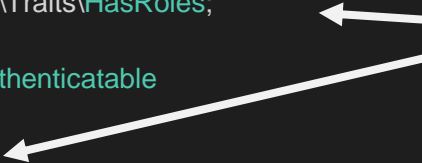
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    use HasRoles;
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
```

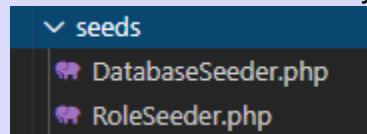


```
'email_verified_at' => 'datetime',
];
}
```

- 4) Selanjutnya kita buat file seeder untuk pembagian level user dengan perintah dibawah ini:

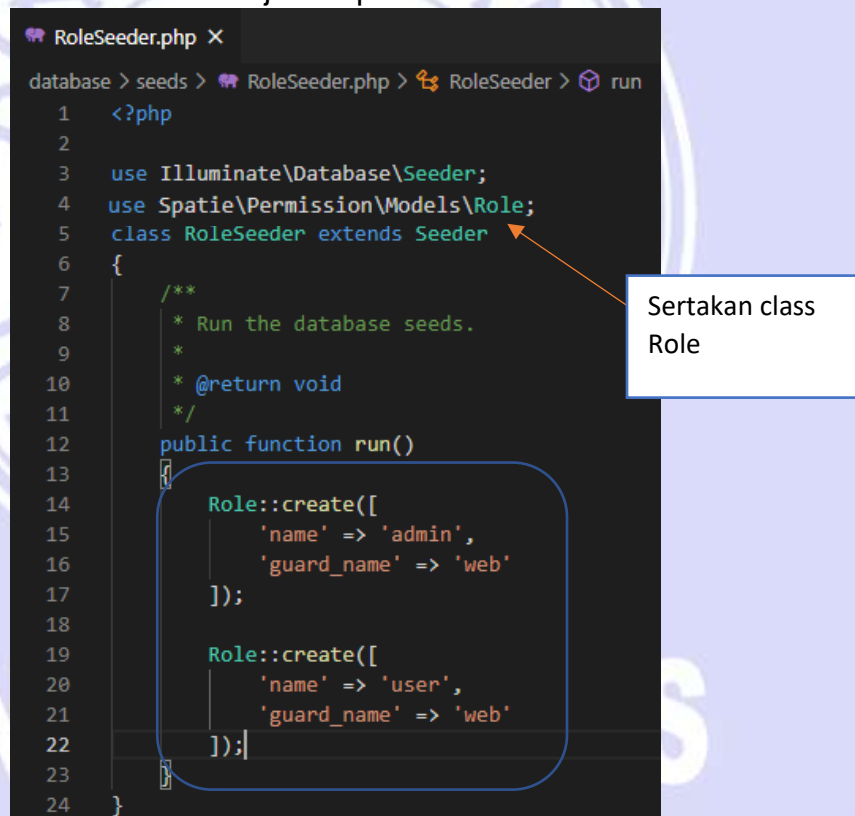
php artisan make:seeder RoleSeeder

maka akan terbentuk filenya didalam folder database/seed sebagai berikut:



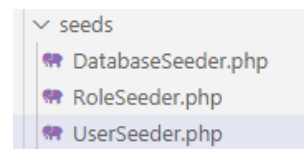
- 5) Langkah selanjutnya kita akan menambah user dengan dua level yakni admin dan user untuk pengelola akses, buka file seeder diatas dan modifikasi kodenya.

Menjadi seperti ini:



- 6) Setelah membuat userRole untuk pengelola akses yakni admin dan user selanjutnya kita buat userSeeder, fungsinya yakni untuk membuat user dan memberikan role/hak akses sebagai berikut:

```
PS C:\xampp\htdocs\ProjectSIA2> php artisan make:seeder UserSeeder
Seeder created successfully.
PS C:\xampp\htdocs\ProjectSIA2> 
```



Buka file UserSeeder lalu tambahkan kode sebagai berikut:

```
use app\User;

public function run()
{
    $admin=User::create([
        'name'=> 'admin',
        'email'=>'admin@test.com',
        'password'=>bcrypt('password')
    ]);
    $admin->assignRole('admin');

    $user=User::create([
        'name'=> 'user',
        'email'=>'user@test.com',
        'password'=>bcrypt('password')
    ]);
    $admin->assignRole('user');
}
```

- 7) Langkah selanjutnya untuk memudahkan proses seeding kita akan panggil class seed yang tadi kita buat kedalam class DatabaseSeeder sebagai berikut:

```
public function run()
{
    // $this->call(UsersTableSeeder::class);
    $this->call(RoleSeeder::class);
    $this->call(UserSeeder::class);
}
```

Perlu diperhatikan dalam memanggil seeder dahulukan RoleSeeder, karena pada UserSeeder ada proses assignRole yang membutuhkan nilai dari data pada tabel roles. Selanjutnya kita jalankan seedernya dengan perintah:

Perintah terminal: php artisan db:seed

apabila menjalankan Seed tidak jalan, maka gunakan perintah dibawah ini:

php artisan migrate:fresh --seed

Jika sudah berhasil akan muncul seperti gambar dibawah ini:

```
Seeding: RoleSeeder
Seeded: RoleSeeder (0.11 seconds)
Seeding: UserSeeder
Seeded: UserSeeder (0.39 seconds)
Database seeding completed successfully.
PS C:\xampp\htdocs\ProjectSIA2>
```

Maka dari hasil seeder diatas yang akan terisi kedalam database diantaranya adalah tabel:

1. Users table:

				id	name	email
<input type="checkbox"/>				1	admin	admin@test.com
<input type="checkbox"/>				2	user	user@test.com

2. Roles table:

				id	name	guard_name	
<input type="checkbox"/>				1	admin	web	2
<input type="checkbox"/>				2	user	web	2

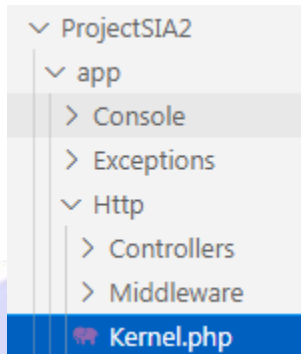
3. Model_has_roles table:

				role_id	model_type	model_id
<input type="checkbox"/>				1	App\User	1
<input type="checkbox"/>				2	App\User	1

Penjelasan mengenai tabel model_has_roles adalah model pivot untuk relasi Many to Many Polymorphic pada Laravel. Intinya roles bisa digunakan dengan relasi ke model lainnya selain model User. Misalkan kita memiliki model Admin kita bisa juga gunakan trait HasRoles di model Admin.

- 8) Pengaturan Route untuk redirect ketika login. Setelah memiliki user yang kita tentukan sesuai dengan levelnya masing-masing maka langkah selanjutnya adalah membuat route akses untuk memisahkan redirect user berdasarkan roles nya masing-masing dalam arti login sebagai admin atau sebagai user biasa. Sebelumnya kita register terlebih dahulu *middlewarenya* dari **spatie/laravel-permission** untuk melindungi route berdasarkan role.

Buka file pada **app/Http/Kernel.php**:



Scroll ke bawah dan cari fungsi **protected \$routeMiddleware = [**
Tambahkan kode dibawah ini:

```
'role' => \Spatie\Permission\Middlewares\RoleMiddleware::class,  
'permission' => \Spatie\Permission\Middlewares\PermissionMiddleware::class,  
'role_or_permission' => \Spatie\Permission\Middlewares\RoleOrPermissionMiddleware::class,
```

Menjadi seperti dibawah ini:

```
protected $routeMiddleware = [  
    'auth' => \App\Http\Middleware\Authenticate::class,  
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
    'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,  
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,  
    'can' => \Illuminate\Auth\Middleware\Authorize::class,  
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,  
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,  
    'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,  
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,  
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,  
    'role' => \Spatie\Permission\Middlewares\RoleMiddleware::class,  
    'permission' => \Spatie\Permission\Middlewares\PermissionMiddleware::class,  
    'role_or_permission' => \Spatie\Permission\Middlewares\RoleOrPermissionMiddleware::class,  
];
```

Jangan lupa save dan close.

Kemudian kita buat **middleware** untuk membatasi akses login antara **Admin** dengan **User**, pertama kita buka file **layout.blade.php** pada folder **views/layouts**.
Tambahkan kode berikut:

@role('admin')

Pernyataan.....

@endrole

Kode lengkapnya sebagai berikut (Baris 51-67)

```
<li class="nav-item">  
    @role('admin')
```



```

<a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapsePages"
  aria-expanded="true" aria-controls="collapsePages">
  <i class="fas fa-fw fa-folder-open"></i>
  <span>Menu utama 1</span>
</a>
<div id="collapsePages" class="collapse" aria-labelledby="headingPages" data-
parent="#accordionSidebar">
  <div class="bg-white py-2 collapse-inner rounded">
    <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 1</a>
    <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 2</a>
    <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 3</a>
    <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 4</a>
  </div>
</div>
@endrole
</li>

```

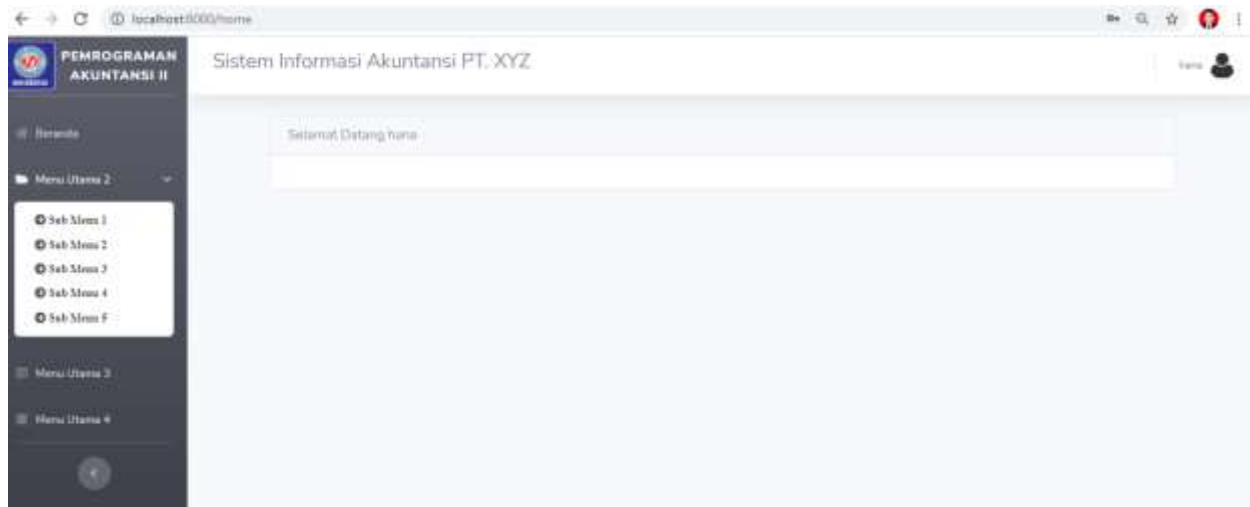
Pada kode diatas kita batasi **menu utama 1 hanya** di akses oleh **Admin** dan **User Biasa** Tidak dapat mengaksesnya. Silakan di save dn jalankan, hasilnya sebagai berikut:

Login sebagai admin



Gambar: login admin full akses

Login sebagai user



Gambar: login sebagai user biasa(akses terbatas)

Jadi pada pembahasan kali ini kita dapat ambil kesimpulan bahwa pada Laravel untuk membentuk sebuah authentication dan permission itu sudah memiliki tool dan modul yang dapat kita manfaatkan seperti pada **Auth dan Permission**.

Untuk menambahkan role otomatis terhadap akses user kita bisa menambahkan fungsi pada file **RegisterController.php** yang berada pada folder **app/http/Controllers/auth/RegisterController.php**, bukalah file tersebut dan modifikasi Menjadi seperti dibawah ini

```
protected function create(array $data)
{
    $user=User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => Hash::make($data['password']),
    ]);
    $user->assignRole('user');
    return $user;
}
```

Sehingga pada saat register user, user sudah mendapatkan **roles** yang di inginkan. Silakan coba register user baru dan lihat hasilnya sebagai berikut:

Tabel user awal:

+ Options

				id	name	email
<input type="checkbox"/>	Edit	Copy	Delete	1	dede	dede.dfs@gmail.com
<input type="checkbox"/>	Edit	Copy	Delete	5	User	user@gmail.com

☐ Check all With selected: Edit Copy

Tabel model_has_role awal:

+ Options

				role_id	model_type	model_id
<input type="checkbox"/>	Edit	Copy	Delete	1	App\User	1
<input type="checkbox"/>	Edit	Copy	Delete	2	App\User	5

Dan sekarang kita coba tambahkan user baru dengan nama: hana dan email: hana@test.com. Berikut hasilnya:

Tabel user setelah register:

+ Options

				id	name	email
<input type="checkbox"/>	Edit	Copy	Delete	1	dede	dede.dfs@gmail.com
<input type="checkbox"/>	Edit	Copy	Delete	5	User	user@gmail.com
<input type="checkbox"/>	Edit	Copy	Delete	14	hana	hana@test.com

Tabel model_has_roles setelah register:

+ Options

				role_id	model_type	model_id
<input type="checkbox"/>	Edit	Copy	Delete	1	App\User	1
<input type="checkbox"/>	Edit	Copy	Delete	2	App\User	5
<input type="checkbox"/>	Edit	Copy	Delete	2	App\User	14

Terlihat bahwa user id 14 mendapatkan hak akses 2(user).

PERTEMUAN 5

3.1 Membuat Form Master User

Untuk mengelola user(Pengguna) kita memerlukan manajemen database melalui form pada laman admin-page. Kita buat tampilan terlebih dahulu, buka visual studio code, dan buat folder dengan nama **admin**, akan terlihat seperti ini **resources/views/admin** klik kanan pada folder admin dan new file, kemudian berikan nama file **user.blade.php** seperti pada gambar berikut:

Selanjutnya buka file **user.blade.php** dan ketikkan source berikut:

```
@extends('layouts.layout')
@section('content')
<div class="d-sm-flex align-items-center justify-content-between mb-4">
    <h1 class="h3 mb-0 text-gray-800">Data Pengguna</h1>
</div><hr>
<div class="card-header py-3 align="right">
    <button class="btn btn-primary btn-sm btn-flat" data-toggle="modal" data-
target="#modal-add"><i
        class="fa fa-plus"></i>Tambah</button>
</div>
<div class="d-sm-flex align-items-center justify-content-between mb-4">
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-bordered" id="dataTable" width="100%"
cellspacing="0">
                <thead>
                    <tr align="center">
                        <th width="5%">User Id</th>
                        <th width="25%">Nama</th>
                        <th width="20%">Email</th>
                        <th width="15%">Roles/Akses</th>
                        <th width="25%">Aksi</th>
                    </tr>
                </thead>
                <tr>
                    @foreach ($user as $row)

                        <td>{{ $row->id }}</td>
                        <td>{{ $row->name }}</td>
                        <td>{{ $row->email }}</td>
                        @foreach ($row->roles as $r)
                            <td>{{ $r->id }}</td>
                        @endforeach
                        <td align="center">
```

```

        <a href="{{route('user.edit' ,[$row->id])}}"
            class="d-none d-sm-inline-block btn btn-sm btn-success shadow-
sm">
            <i class="fas fa-edit fa-sm text-white-50"></i>Edit Akses
        </a>
        <a href="/user/hapus/{{ $row->id }}" onclick="return confirm('Yakin
Ingin menghapus data?')" class="d-none d-sm-inline-block btn btn-sm btn-danger
shadow-sm">
            <i class="fas fa-trash-alt fa-sm text-white-50"></i> Hapus
        </a>
    </td>
</tr>
</tbody>
</table>
</div>
</div>
</div>
<!-- modal add data-->
<div class="modal inmodal fade" id="modal-add" tabindex="-1" role="dialog" aria-
hidden="true">
    <div class="modal-dialog modal-xs">
        <form name="frm_add" id="frm_add" class="form-horizontal" action="#"
method="POST"
            enctype="multipart/form-data">
            @csrf
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Tambah Data User</h4>
                </div>
                <div class="modal-body">
                    <div class="form-group"><label class="col-lg-20 control-
label">Nama User</label>
                        <div class="col-lg-10"><input type="text"
name="username" required
                            class="form-control"></div>
                    <div class="form-group"><label class="col-lg-20 control-
label">Email User</label>
                        <div class="col-lg-10"><input type="email"
name="email" required
                            class="form-control"></div>
                    <div class="form-group"><label class="col-lg-20
control-label">Roles/Akses</label>
                        <div class="col-lg-10">

```



```

                                <select id="roles" name="roles" class="form-
control" required>
                                <option value="">--Pilih Roles--</option>
                                <option value="admin">Admin</option>
                                <option value="user">User</option>
                                </select>
                                </div>
                                </div>
                                </div>
                                <div class="modal-footer">
                                <button type="button" class="btn btn-light" data-
dismiss="modal">Tutup</button>
                                <button type="submit" class="btn btn-
primary">Simpan</button>
                                </div>
                                @endsection
                        </div>
                </form>
        </div>
</div>

```

Kemudian pada web route tambahkan route sebagai berikut:

```

//User
Route:: resource('/user','UserController' );
Route:: get('/user/hapus/{id}' , 'UserController@destroy' );

```

Apabila belum membuat User Controller, silahkan buat dengan perintah dibawah ini:

php artisan make:controller UserController --resource

Jika perintah sudah berhasil, selanjutnya untuk menampilkannya buka file **UserController.php** dan tambahkan perintah yang di garis merah seperti dibawah ini:

```

<?php
namespace App\Http\Controllers;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
Use alert;
use RealRashid\SweetAlert\Facades\Alert as FacadesAlert;
use Spatie\Permission\Models\Role;
class UserController extends Controller
{

```

```

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function index()
{
    $user=\App\User::all();
    return view('admin.user.user',['user'=>$user]);
}
/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    //
}
/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $save_user= new \App\User;
    $save_user->name=$request->get('username');
    $save_user->email=$request->get('email');
    $save_user->password=bcrypt('password');
    if ($request->get('roles')== 'ADMIN') {
        # code...
        $save_user->assignRole('admin');
    } else {
        # code...
        $save_user->assignRole('user');
    }
    $save_user->save();

    return redirect()->route('user.index');
}
/**
 * Display the specified resource.

```

```

*
* @param int $id
* @return \Illuminate\Http\Response
*/
public function show($id)
{
    $user = User::find($id);
    return view('admin.user.user',compact('user'));
}
/**
 * Show the form for editing the specified resource.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    $user = User::find($id);
    $roles = Role::pluck('name')->all();
    $userRole = $user->roles->pluck('name')->all();
    return view('admin.user.editUser',compact('user','roles','userRole'));
}
/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $user = User::find($id);
    DB::table('model_has_roles')->where('model_id',$id)->delete();
    $user->assignRole($request->input('role'));

    return redirect()->route( 'user.index');
}
/**
 * Remove the specified resource from storage.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)

```

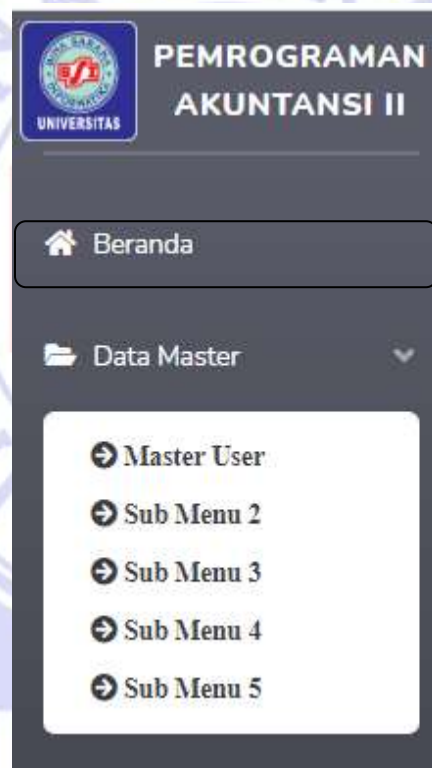
```

{
    $hapus = \App\User::findOrFail($id);
    $hapus->delete();
    $hapus->removeRole('admin','user');
    return redirect()->route('user.index');
}
}

```

use Alert yaitu untuk memanggil Library Alert dan dalam kotak merah kedua yaitu untuk passing/kirim data Controller ke dalam view, sehingga data dalam tabel user akan di tampilkan pada form user.

Selanjutnya, untuk menempatkan pada bagian admin kita rubah template layout pada admin layout di bagian menu 1 dan sub menu 1 menjadi sebagai berikut:



Tampilan dashboard menu admin

Source file layout.blade.php(Baris 43-66)

```

<!-- Divider -->
<hr class="sidebar-divider">
<li class="nav-item">
    <a class="nav-link" href="{{ route('home') }}">
        <i class="fas fa-fw fa-home"></i>
        <span>Beranda</span></a>
</li>

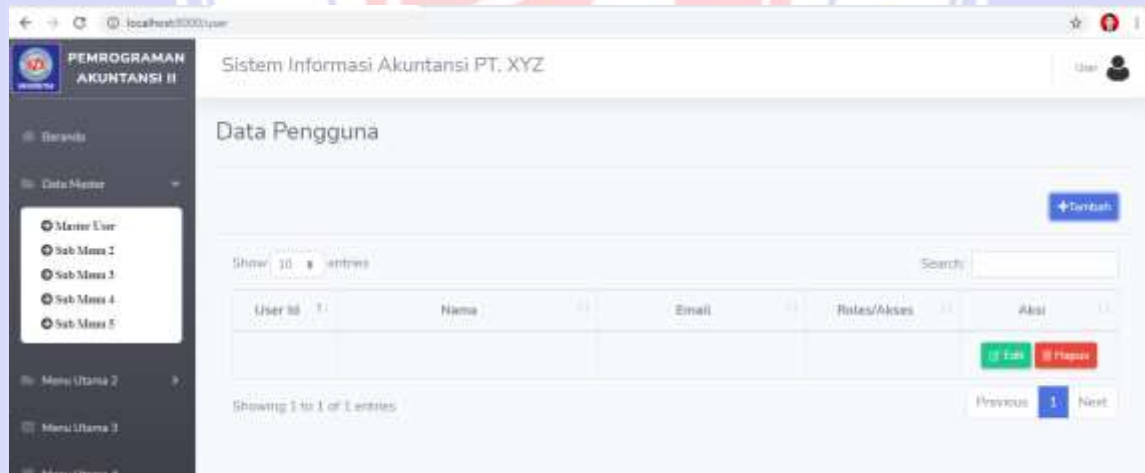
```

```

<!-- Nav Item - Pages Collapse Menu -->
<li class="nav-item">
  <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapsePages"
    aria-expanded="true" aria-controls="collapsePages">
    <i class="fas fa-fw fa-folder-open"></i>
    <span>Data Master</span>
  </a>
  <div id="collapsePages" class="collapse" aria-labelledby="headingPages" data-
parent="#accordionSidebar">
    <div class="bg-white py-2 collapse-inner rounded">
      <a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('user.index') }}"> Master User</a>
      <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 2</a>
      <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 3</a>
      <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 4</a>
      <a class="collapse-item fas fa-arrow-circle-right" href="#"> Sub Menu 5</a>
    </div>
  </div>
</li>

```

Kemudian loginlah sebagai admin dan akses menu master, tampilan sebagai berikut:



Selanjutnya kita akan membuat input user, silahkan tambahkan pada Folder User dengan nama **input.blade.php**, tambahkan Code dibawah ini:

```

@extends('layouts.layout')
@section('content')
<form action="{{route('user.store')}}" method="POST">@csrf
  <fieldset><legend>InputDataPengguna</legend>
  <div class="form-group row">
    <div class="col-md-5">
      <label for="email">Email</label>

```

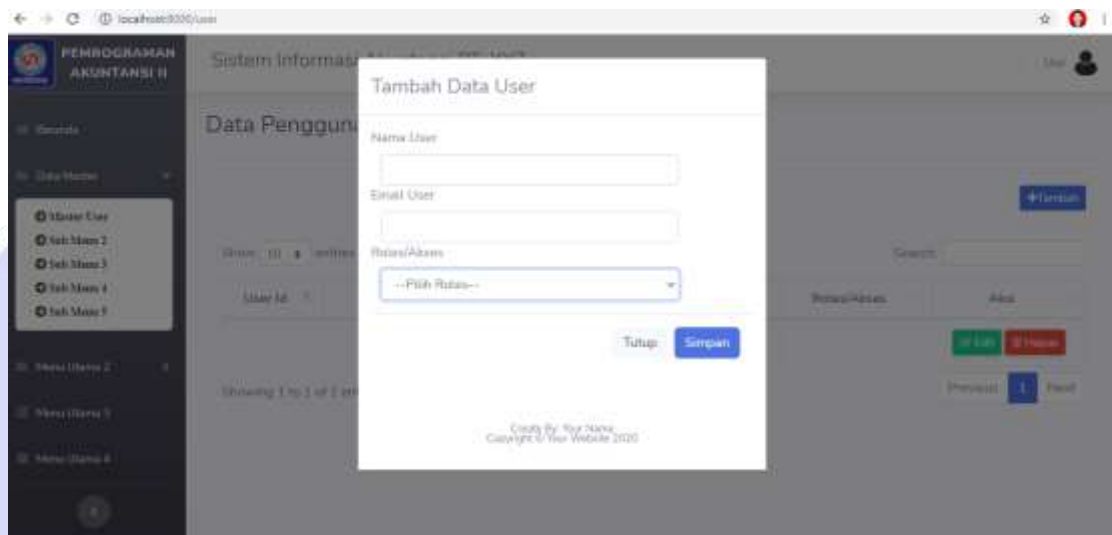
```

        <input id="email" type="email" name="email" class="form-control"
required>
    </div>
</div>
    <div class="col-md-5">
        <label for="nama">Nama Lengkap</label>
        <input id="nama" type="text" name="nama" class="form-
control"required>
    </div>
</div>
<div class="form-group row">
    <div class="col-md-5">
        <label for="roles">Roles</label>
        <select id="roles" name="roles[]" class="form-control" required>
            <option value="">--PilihRoles--</option>
            <option value="admin">Admin</option>
            <option value="user">User</option>
        </select>
    </div>

    <div class="form-group row">
        <div class="col-md-5">
            <label for="passw">Password</label>
            <input id="passw" type="password" name="passw" class="form-
control"required>
        </div>
    </div>
    <div class="col-md-10">
        <input type="submit" class="btn btn-success btn-send" value="Simpan">
        <input type="Button" class="btn btn-primary btn-send" value="Kembali"
onclick="history.go(-1)">
    </div>
    <hr>
</fieldset>
</form>
@endsection

```


Tampilan tambah data user:



Selanjutnya membuat form edit user buatlah blade baru pada folder **views** dengan nama **editUser.blade.php** dan berikanlah kode html sebagai berikut:

```
@extends('layouts.layout')
@section('content')
<form action="{{route('user.update', [$user->id])}}" method="POST">
    @csrf
    <input type="hidden" name="_method" value="PUT">
    <fieldset>
        <legend>Ubah Akses User</legend>
        <div class="form-group row">
            <div class="col-md-2">
                <label for="kode">Kode User</label>
                <input class="form-control" type="text" name="kode"
value="{{${user->id}}}" readonly>
            </div>
            <div class="col-md-5">
                <label for="user">Nama User</label>
                <input id="name" type="text" name="uname" class="form-control"
value="{{${user->name}}}" readonly>
            </div>
        </div>
        <div class="form-group row">
            <div class="col-md-5">
                <label for="email">Email</label>
                <input id="email" type="text" name="email" class="form-control"
value="{{${user->email}}}" readonly>
            </div>
            <div class="col-md-2">
```

```

        @foreach ($user ->roles as $role)
        <label for="akses">Akses</label>
        <input id="akses" type="text" name="akses" class="form-control"
value="{{ $role->id }}" readonly>
        @endforeach
    </div>
</div>
<div class="form-group row">
<div class="col-md-2">
    <label for="akses">Ubah Akses</label>
    <select id="roles" name="role" class="form-control" required>
        <option value="">--Pilih Akses--</option>
        <option value="admin">Admin</option>
        <option value="user">User</option>
    </select>

</div>

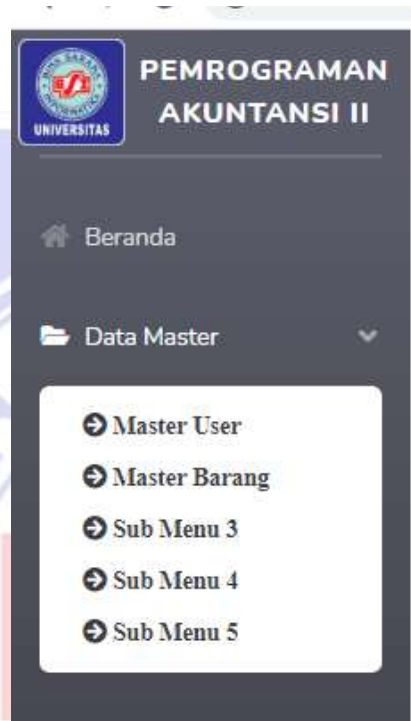
</fieldset>
<div class="col-md-10">
    <input type="submit" class="btn btn-success btn-send" value="Ubah Akses">
    <a href="{{ route('user.index') }}"><input type="Button" class="btn btn-
primary btn-send" value="Kembali"></a>
</div>
<hr>
</form>
@endsection

```

Dan jalankan perintah: **php artisan serve** dan lakukanlah update ke nama user **user** menjadi **admin** hasilnya sebagai berikut:

PERTEMUAN 6

6.1. Membuat Master Data Barang

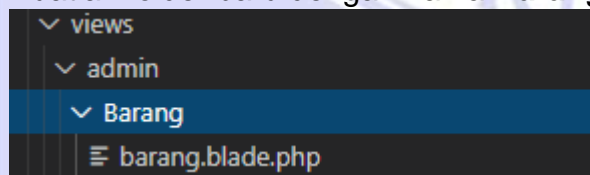


Gambar Navigasi Master Barang

Untuk keperluan master data selanjutnya kita buat data barang sebagai bahan untuk melakukan transaksi, berikut tahapannya:

1. Membuat tampilan form data barang

Buatlah folder baru dengan nama **Barang** pada **view/admin/**,



kemudian Buat file baru dengan nama **barang.blade.php** pada folder **view/admin/Barang/barang.blade.php** berikut source formnya:

```
@extends('layouts.layout')
@section('content')
<div class="d-sm-flex align-items-center justify-content-between mb-4">
  <h1 class="h3 mb-0 text-gray-800">Data Barang</h1>
</div>
<hr>
<div class="card-header py-3 align="right">
```

```

<button type="button" class="d-none d-sm-inline-block btn btn-sm btn-
primary shadow-sm" data-toggle="modal" data-target="#exampleModalScrollable">
<i class="fas fa-plus fa-sm text-white-50"></i> Tambah
</button>
</div>
<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">
<div class="table-responsive">
<table class="table table-bordered table-
striped" id="dataTable" width="100%" cellpadding="0">
  <thead class="thead-dark">
    <tr>
      <th>Kode Barang</th>
      <th>Nama Barang</th>
      <th>Harga Barang</th>
      <th>Stok Barang</th>
      <th>Aksi</th>
    </tr>
  </thead>
  <tbody>
    @foreach($barang as $brg)
      <tr>
        <td>{{ $brg->kd_brg}}</td>
        <td>{{ $brg->nm_brg}}</td>
        <td>{{ number_format($brg->harga)}}</td>
        <td>{{ number_format($brg->stok)}}</td>
        <td align="center"><a href="{{route('barang.edit',[$brg-
>kd_brg])}}" class="d-none d-sm-inline-block btn btn-sm btn-success shadow-sm">
          <i class="fas fa-edit fa-sm text-white-50"></i> Edit</a>
          <a href="/barang/hapus/{{ $brg-
>kd_brg}}" onclick="return confirm('Yakin Ingin menghapus data?')" class="d-
none d-sm-inline-block btn btn-sm btn-danger shadow-sm">
            <i class="fas fa-trash-alt fa-sm text-white-50"></i> Hapus</a>
          </td>
        </tr>
      @endforeach
    </tbody>
  </table>
</div>
</div>
</div>
<div class="modal fade" id="exampleModalScrollable" tabindex="-1" role="dialog"
aria-labelledby="exampleModalScrollableTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-scrollable" role="document">

```

```

<div class="modal-content">
  <div class="modal-header">
    <h5 class="modal-title" id="exampleModalScrollableTitle">Tambah Data
Barang</h5>
    <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <form action="{ { action('barangController@store') } }" method="POST">
    @csrf
    <div class="modal-body">
      <div class="form-group">
        <label for="exampleFormControlInput1">Kode Barang</label>
        <input type="text" name="addkdbrg" id="addkdbrg" class="form-control"
maxlength="5" id="exampleFormControlInput1" >
      </div>
      <div class="form-group">
        <label for="exampleFormControlInput1">Nama Barang</label>
        <input type="text" name="addnmbrg" id="addnmbrg" class="form-control"
id="exampleFormControlInput1" >
      </div>
      <div class="form-group">
        <label for="exampleFormControlInput1">Harga Barang</label>
        <input type="number" name="addharga" id="addharga" class="form-
control" id="exampleFormControlInput1" >
      </div>
      <div class="form-group">
        <label for="exampleFormControlInput1">Stok Barang</label>
        <input type="number" name="addstok" id="addstok" class="form-control"
id="exampleFormControlInput1" >
      </div>
    </div>
    <div class="modal-footer">
      <button type="button" class="btn btn-secondary" data-dismiss="modal">
Batal</button>
      <input type="submit" class="btn btn-primary btn-send" value="Simpan">
    </div>
  </div>
</form>
</div>
@endsection

```

2. Membuat model barang

Langkah selanjutnya, tambahkan model, akan tetapi sudah dilakukan di pertemuan pertama, maka silahkan buka model barang pada folder **app/Barang.php** tambahkan source code sebagai berikut:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Barang extends Model
{
    //jika tidak di definisikan, maka primary akan terdetek id
    protected $primaryKey = 'kd_brg';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "barang";
    protected $fillable=['kd_brg','nm_brg','harga','stok'];
}
```

3. Pembuatan Controller Barang

Kemudian untuk pembuatan controller yang dibutuhkan dalam pengisian method-method dalam pengelolaan CRUD dengan perintah: **php artisan make:controller barangController --resource**, bukalah file **barangController** pada folder **app/http/controller** dan tambahkan code sebagai berikut:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Alert;
class BarangController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $barang=\App\Barang::All();
        return view('admin.barang.barang',['barang'=>$barang]);
    }
    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
```



```

{
    //
    return view('admin.barang.input');
}
/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
    $tambah_barang=new \App\Barang;
    $tambah_barang->kd_brg = $request->addkdbrg;
    $tambah_barang->nm_brg = $request->addnmbrg;
    $tambah_barang->harga = $request->addharga;
    $tambah_barang->stok = $request->addstok;
    $tambah_barang->save();
    Alert::success('Pesan ', 'Data berhasil disimpan');
    return redirect('/barang');
}

public function edit($id)
{
    $barang_edit=\App\Barang::findOrFail($id);
    return view('admin.barang.editBarang', ['barang'=>$barang_edit]);
}

public function update(Request $request, $id)
{
    $barang=\App\Barang::findOrFail($id);
    $barang->kd_brg=$request->get('addkdbrg');
    $barang->nm_brg=$request->get('addnmbrg');
    $barang->harga=$request->get('addharga');
    $barang->stok=$request->get('addstok');
    $barang->save();
    return redirect()->route('barang.index');
}

```

```

public function destroy($kd_brg)
{
    //
    $barang=\App\Barang::findOrFail($id);
    $barang->delete();
    Alert::success('Pesan ', 'Data berhasil dihapus');
    return redirect()->route('barang.index');
}

```

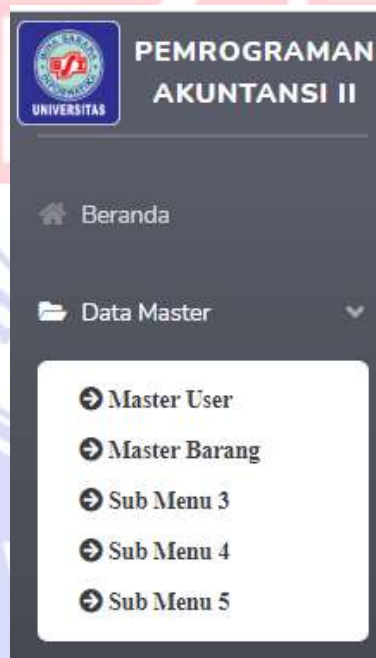
Untuk menentukan rute/jalur akses form barang kita tambahkan route dibawah ini pada file **web.php** pada folder **routes/web**

```

Route::resource('/barang','barangController');
Route::get('/barang/hapus/{id}','barangController@destroy');

```

Selanjutnya kita atur navigasi pada layout admin agar menjadi seperti pada gamabar berikut ini:



Gambar Navigasi Master Data Barang

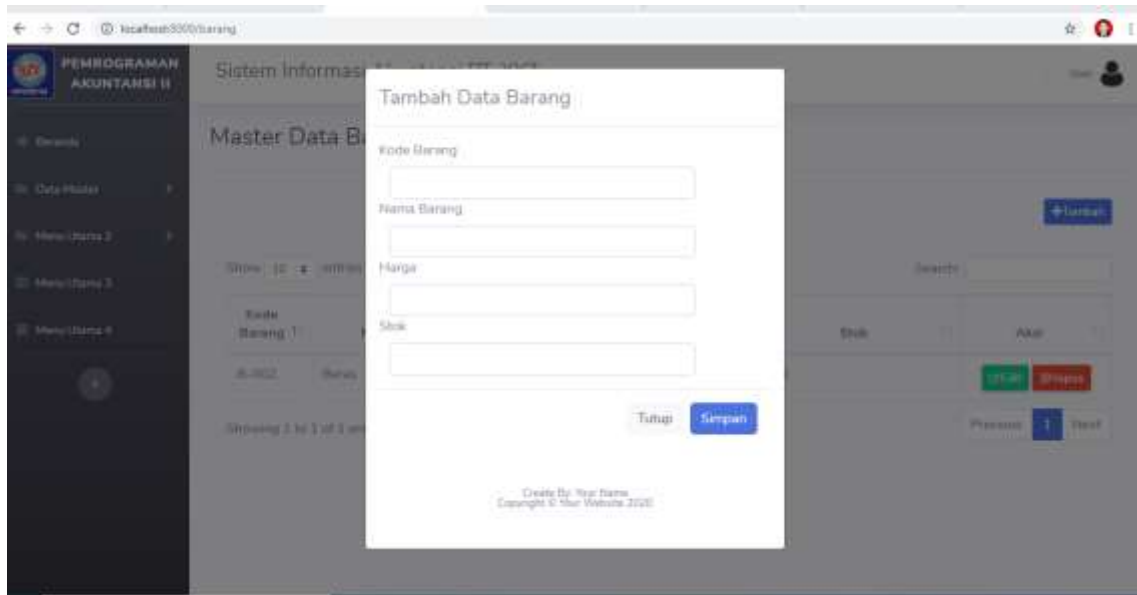
Langkahnya mudah, bukalah file **layout.blade.php** kemudian rubah pada sub menu 2 menjadi seperti berikut(baris ke-60 atau sesuaikan dengan sub menu 2 yang diganti):

```

<a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('barang.index') }}"> Master Barang</a>

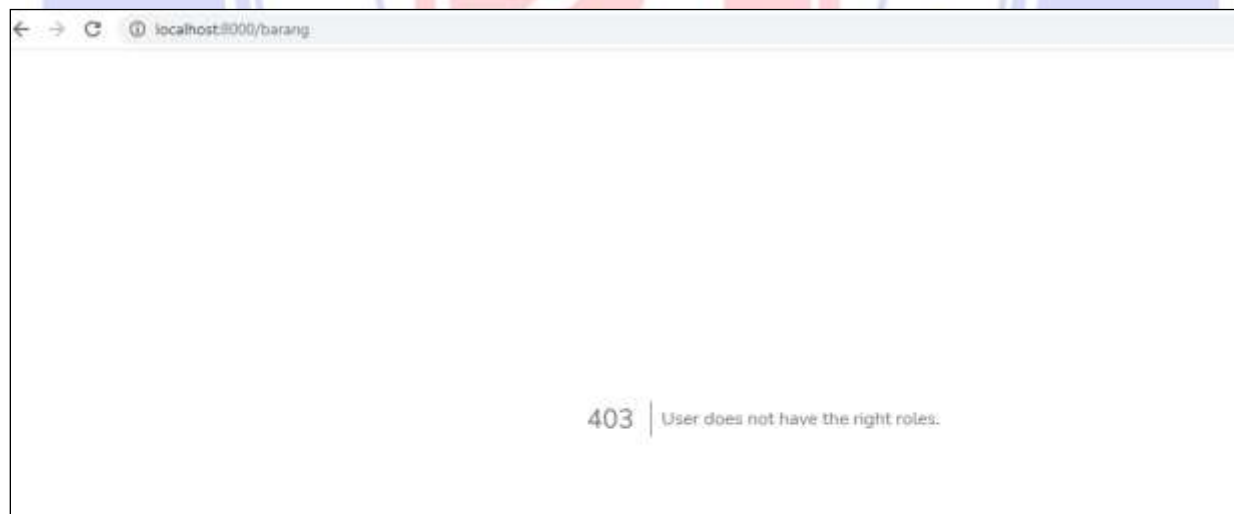
```

Untuk hasilnya silakan jalankan dengan perintah di terminal: **php artisan serve**



Gambar: Form Master Barang

Selanjutnya untuk menjaga agar hak akses tetap di akses oleh admin, kita sudah membuat group middleware admin pada route barang, Sehingga ketika user biasa bukan admin mencoba mengakses page barang maka yang terjadi adalah sebagai berikut:



Untuk mencobanya silakan login sebagai user biasa, kemudian pada url ketikkan alamat berikut: <http://localhost:8000/barang>.

4. Membuat Edit Data Barang

Selanjutnya untuk mengelola update data maka kita perlu menambahkan form edit maka kita tambahkan file dengan nama: **editBarang.blade.php** pada folder **views/admin/Barang**

```
@extends('layouts.layout')
@section('content')
```

```

<form action="{{route('barang.update', [$barang->kd_brg])}}" method="POST">
    @csrf
    <input type="hidden" name="_method" value="PUT">
    <fieldset>
        <legend>Ubah Data Barang</legend>
        <div class="form-group row">
            <div class="col-md-5">
                <label for="addkdbrg">Kode Barang</label>
                <input class="form-
control" type="text" name="addkdbrg" value="{{ $barang->kd_brg }}" readonly>
            </div>
            <div class="col-md-5">
                <label for="addnmbrg">Nama Barang</label>
                <input id="addnmbrg" type="text" name="addnmbrg" class="form-
control" value="{{ $barang->nm_brg }}">
            </div>
        </div>
        <div class="form-group row">
            <div class="col-md-5">
                <label for="Harga">Harga</label>
                <input id="addharga" type="text" name="addharga" class="form-
control" value="{{ $barang->harga }}">
            </div>
            <div class="col-md-5">
                <label for="Stok">Stok</label>
                <input id="addstok" type="text" name="addstok" class="form-
control" value="{{ $barang->stok }}">
            </div>
        </div>
    </fieldset>
    <div class="col-md-10">
        <input type="submit" class="btn btn-success btn-send" value="Update">
        <a href="{{ route('barang.index') }}"><input type="Button" class="btn btn-
primary btn-send" value="Kembali"></a>
    </div>
    <hr>
</form>
@endsection

```

Untuk mendapatkan link route kita tambahkan url route pada form **barang.blade.php** dengan source sebagai berikut pada baris tombol edit(baris ke 32):

```

31         <td align="center">
32             <a href="#"
33                 class="d-none d-sm-inline-block btn btn-sm btn-success shadow-sm">
34                 <i class="fas fa-edit fa-sm text-white-50"></i>Edit
35             </a>

```

```
{{route('barang.edit',[$brg->kd_brg])}}
```

Maka akan tampil seperti gambar dibawah ini

```

<td>{{ $brg->kd_brg}}</td>
<td>{{ $brg->nm_brg}}</td>
<td>{{ number_format($brg->harga)}}</td>
<td>{{ number_format($brg->stok)}}</td>
<td align="center"><a href="{{route('barang.edit',[$brg->kd_brg])}}" class="d-none d-sm-in
<i class="fas fa-edit fa-sm text-white-50"></i> Edit</a>
<a href="/barang/hapus/{{ $brg->kd_brg }}" onclick="return confirm('Yakin Ingin menghapus d
<i class="fas fa-trash-alt fa-sm text-white-50"></i> Hapus</a>
</td>

```

Apabila sudah sesuai dengan gambar diatas, silahkan coba jalankan ke form Master Barang, lalu nanti klik tombol **Edit**, maka akan tampil form seperti dibawah ini:

Gambar: Form Merubah Data Barang

TUGAS 2:

1. Buatlah form master supplier beserta CRUD dan permission dengan ketentuan seperti form barang, berikut tampilan form master supplier



Gambar: Tampilan Struktur Navigasi Master Supplier



Gambar: Form Tambah Data Supplier



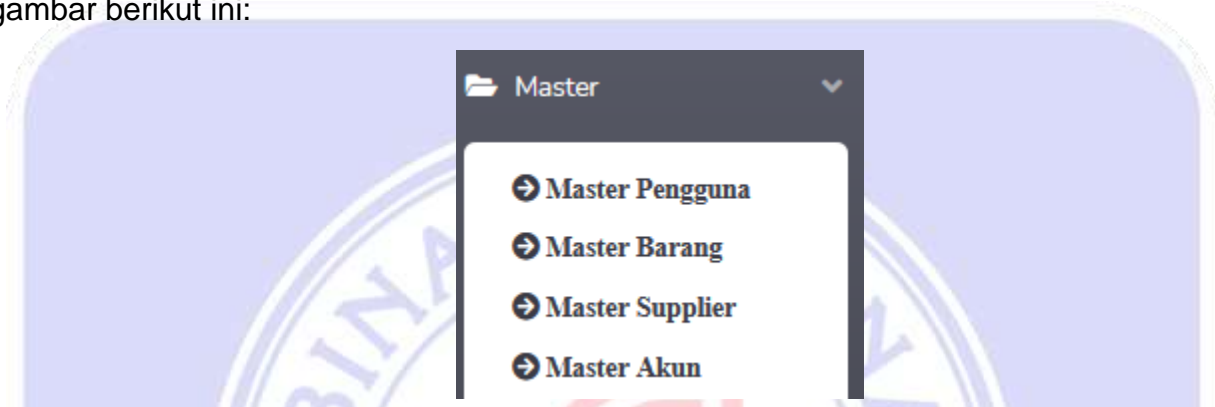
Gambar: Form Edit Data Supplier

PERTEMUAN 7

7.1. Form Pengelola Data Master Akun

1. Setting Navigasi Master Akun

Selanjutnya kita atur navigasi pada layout admin agar menjadi seperti pada gambar berikut ini:



Gambar Navigasi Master Data Akun

Langkahnya mudah, bukalah file **layout.blade.php** kemudian rubah pada sub menu 4 menjadi seperti berikut:

```
<a class="collapse-item fas fa-arrow-circle-  
right" href="{{ route('akun.index') }}"> Master Akun</a>
```

2. Membuat model akun

Langkah selanjutnya, tambahkan model, akan tetapi sudah dilakukan di pertemuan pertama, maka silahkan buka model akun pada folder **app/Akun.php** tambahkan source code sebagai berikut:

```
<?php  
namespace App;  
use Illuminate\Database\Eloquent\Model;  
class Akun extends Model  
{  
    protected $primaryKey = 'no_akun';  
    public $incrementing = false;  
    protected $keyType = 'string';  
    public $timestamps = false;  
    protected $table = "akun";  
    protected $fillable=['no_akun','nama_akun'];  
}
```

3. Membuat Controller Akun

Buat controller resource baru dengan nama Akun dengan menggunakan terminal dengan perintah dibawah ini

php artisan make:controller AkunController --resource

Setelah itu atur route dengan membuka file web.php kemudian tambahkan kode seperti dibawah ini.

```
Route::resource('/akun', 'AkunController');
```

4. Membuat tampilan halaman data akun

Buat folder baru pada folder admin dengan nama akun, seperti data master barang, master supplier, kemudian buat file pada file tersebut dengan cara klik kanan pada folder akun lalu pilih New File lalu berikan nama file **akun.blade.php** kemudian ketikkan kode seperti dibawah ini

```
@extends('layouts.layout')
@section('content')
@include('sweetalert::alert')
<div class="d-sm-flex align-items-center justify-content-between mb-4">
    <h1 class="h3 mb-0 text-gray-800">Data Akun Rekening</h1>
</div>
<hr>
<div class="card-header py-3 align="right">
<button type="button" class="d-none d-sm-inline-block btn btn-sm btn-
primary shadow-sm" data-toggle="modal" data-target="#exampleModalScrollable">
<i class="fas fa-plus fa-sm text-white-50"></i> Tambah
</button>
</div>
<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">
<div class="table-responsive">
<table class="table table-bordered table-
striped " id="dataTable" width="100%" cellpadding="0">
    <thead class="thead-dark">
        <tr>
            <th width="20%">KodeAkun</th>
            <th>NamaAkun</th>
            <th width="20%">Aksi</th>
        </tr>
    </thead>
    <tbody>
        @foreach($akun as $akn)
        <tr>
            <td>{{ $akn->no_akun }}</td>
            <td>{{ $akn->nm_akun }}</td>
            <td align="center"><a href="{{ route('akun.edit', [$akn-
no_akun]) }}" class="d-none d-sm-inline-block btn btn-sm btn-success shadow-sm">
                <i class="fas fa-edit fa-sm text-white-50"></i> Edit</a>
            </td>
        </tr>
        </tbody>
    </table>
    </div>
</div>
</div>
```

```

        <a href="/akun/hapus/{{ $akn-
>no_akun }}" onclick="return confirm('Yakin Ingin menghapus data?')" class="d-
none d-sm-inline-block btn btn-sm btn-danger shadow-sm">
        <i class="fas fa-trash-alt fa-sm text-white-50"></i> Hapus</a>
    </td>
</tr>
</tbody>
</table>
</div>
</div>
</div>

<div class="modal fade" id="exampleModalScrollable" tabindex="-
1" role="dialog" aria-labelledby="exampleModalScrollableTitle" aria-
hidden="true">
    <div class="modal-dialog modal-dialog-scrollable" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-
title" id="exampleModalScrollableTitle">Tambah Data Akun</h5>
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <form action="{{ action('AkunController@store') }}" method="POST">
                @csrf
                <div class="modal-body">
                    <div class="form-group">
                        <label for="exampleFormControlInput1">Kode Akun</label>
                        <input type="text" name="addnoakun" id="addnoakun" class="form-
control" id="exampleFormControlInput1" >
                    </div>
                    <div class="form-group">
                        <label for="exampleFormControlInput1">Nama Akun</label>
                        <input type="text" name="addnmakun" id="addnmakun" class="form-
control" id="exampleFormControlInput1" >
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-
dismiss="modal"> Batal</button>
                    <input type="submit" class="btn btn-primary btn-send" value="Simpan">
                </div>
            </form>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</form>
</div>
</div>
@endsection

```

Untuk menampilkan data yang terdapat pada tabel akun pada halaman index akun, tambahkan kode untuk mengambil data pada file **AkunController.php**, pada action index tuliskan kode berikut ini.

```

public function index()
{
    $akun=\App\Akun::All();
    return view('admin.akun.akun', ['akun'=>$akun]); //Array
}

```

5. Membuat tampilan input data akun.

Buat file baru dengan cara klik kanan pada folder Akun lalu pilih New File lalu berikan nama file **input.blade.php** kemudian ketikkan kode seperti dibawah ini

```

@extends('layouts.layout')
@section('content')
<form action="{{route('akun.store')}}" method="POST">@csrf
    <fieldset><legend>Input Data Akun</legend>
    <div class="form-group row">
        <div class="col-md-5">
            <label for="usname">Kode Akun</label>
            <input id="addnoakun" type="text" name="addnoakun" class="form-
control"required>
        </div>
        <div class="col-md-5">
            <label for="nama">Nama Akun</label>
            <input id="addnmakun" type="text" name="addnmakun" class="form-
control"required>
        </div>
    </div>
    <div class="col-md-10">
        <input type="submit" class="btn btn-success btn-send" value="Simpan">
        <input type="Button" class="btn btn-primary btn-
send" value="Kembali" onclick="history.go(-1)">
    </div>
    <hr>
</fieldset>
</form>
@endsection

```

Untuk menampilkan halaman input data akun, perlu ditambahkan kode pada file **AkunController.php** pada action create dengan menuliskan kode berikut ini.

```
public function create()
{
    return view('admin.akun.input');
}
```



Gambar: Tampil Form Data Akun

Untuk menyimpan data yang diinput pada form input data akun, buka file **AkunController.php** pada action store masukan kode seperti dibawah ini

```
public function store(Request $request) // fungsi
{
    //untuk menyimpan data
    $tambah_akun=new \App\Akun;
    $tambah_akun->no_akun = $request->addnoakun;
    $tambah_akun->nm_akun = $request->addnmakun;
    $tambah_akun->save(); // method
    return redirect('/akun'); // prosedur
}
```

Silahkan inputkan data akun di bawah ini:

Kode Akun	Nama Akun
101	Kas
500	Pembelian
501	Retur

6. Membuat form ubah data akun

Buat file baru dengan cara klik kanan pada folder akun lalu pilih New File lalu berikan nama file **edit.blade.php** kemudian ketikkan kode seperti dibawah ini.

```
@extends('layouts.layout')
@section('content')
<form action="{{route('akun.update',[$akun->no_akun])}}" method="POST">@csrf
```

```

<input type="hidden" name="_method" value="PUT">
<fieldset><legend>Rubah Form Data Akun</legend>
  <div class="form-group row">
    <div class="col-md-5">
      <label for="addnoakun">Kode Akun</label>
      <input id="addnoakun" type="text" name="addnoakun" class="form-
control" value="{{ $akun->no_akun }}">
    </div>
    <div class="col-md-5">
      <label for="addnmakun">Nama Akun</label>
      <input id="addnmakun" type="text" name="addnmakun" class="form-
control" value="{{ $akun->nm_akun }}"></div>
    </div>

    <div class="col-md-10">
      <input type="submit" class="btn btn-success btn-send" value="Update">
      <a href="{{ route('akun.index') }}"><input type="Button" class="btn btn-
primary btn-send" value="Kembali"></a>
    </div>
  <hr>
</fieldset>
</form>
@endsection

```

Untuk menyimpan perubahan data form edit data akun, buka file **AkunController.php** pada action update masukan kode seperti dibawah ini.

```

public function edit($id)
{
    $akun_edit=\App\Akun::findOrFail($id);
    return view('admin.akun.edit',['akun'=>$akun_edit]);
}

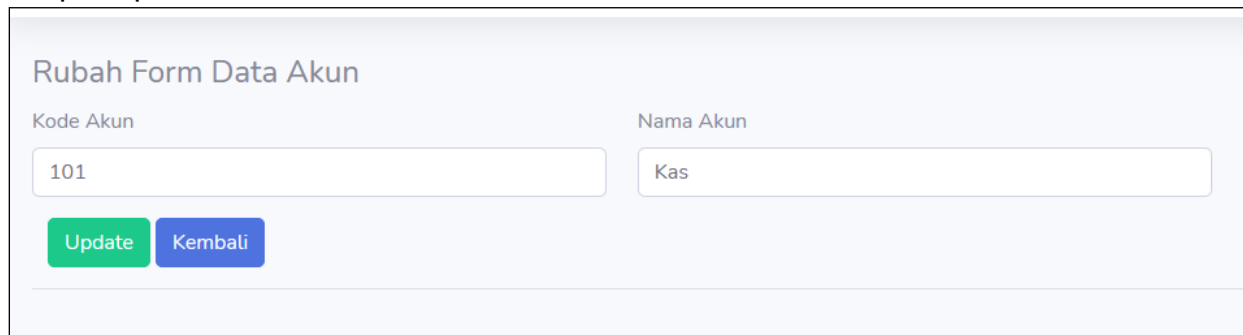
public function update(Request $request, $id)
{
    $update_akun = \App\akun::findOrFail($id);
    $update_akun->no_akun=$request->addnoakun;
    $update_akun->nm_akun=$request->addnmakun;
    $update_akun->save();
    Alert::success('Update', 'Data Berhasil di update');
    return redirect()->route( 'akun.index');
    $tambah_akun->no_akun = $request->addnoakun;
}

```

Setelah itu perlu menambahkan kode pada file route **web.php** seperti dibawah ini.

```
Route::get('/akun/edit/{id}','AkunController@edit');
```


Untuk melihat hasil luarannya klik menu Edit pada halaman data akun, maka akan tampil seperti dibawah ini.



7. Menghapus data akun

Untuk menghapus data akun buka file **AkunController.php** kemudian tambahkan kode dibawah ini pada action destroy.

```
public function destroy($no_akun)
{
    $akun=\App\Akun::findOrFail($no_akun);
    $akun->delete();
    Alert::success('Pesan ', 'Data berhasil dihapus');
    return redirect()->route('akun.index');
}
```

Setelah itu perlu menambahkan kode pada file route **web.php** seperti dibawah ini.

```
Route::get('/akun/hapus/{id}', 'AkunController@destroy');
```

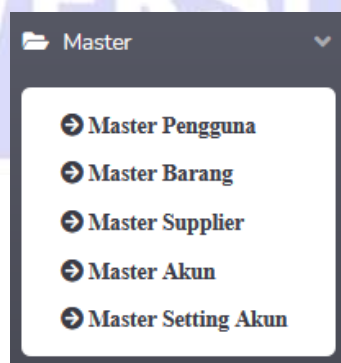
Maka route ada menjadi 3 seperti gambar dibawah ini

```
## Data Akun
Route::resource('/akun', 'AkunController')->middleware('role:admin'); //Tampil
Route::get('/akun/hapus/{kode}', 'AkunController@destroy'); // Hapus
Route::get('/akun/edit/{id}', 'AkunController@edit'); //Edit
```

7.2. Membuat Master Setting Akun

1. Setting Navigasi Master Setting

Selanjutnya kita atur navigasi pada layout admin agar menjadi seperti pada gambar berikut ini:



Gambar Navigasi Master Data Setting Akun

Langkahnya mudah, bukalah file **layout.blade.php** kemudian rubah pada sub menu 4 menjadi seperti berikut:

```
<a class="collapse-item fas fa-arrow-circle-  
right" href="{{ route('setting.transaksi') }}"> Master Setting Akun</a>
```

2. Membuat model Setting

Langkah selanjutnya, tambahkan model, akan tetapi sudah dilakukan di pertemuan pertama, maka silahkan buka model akun pada folder **app/Setting.php** tambahkan source code sebagai berikut:

```
<?php  
namespace App;  
use Illuminate\Database\Eloquent\Model;  
class Setting extends Model  
{  
    protected $primaryKey = 'id_setting';  
    public $incrementing = false;  
    protected $keyType = 'string';  
    public $timestamps = false;  
    protected $table = "setting";  
    protected $fillable=['id_setting','no_akun','nama_transaksi'];  
}
```

3. Membuat Controller Setting Akun

Buat controller resource baru dengan nama Setting Akun dengan menggunakan terminal dengan perintah dibawah ini

php artisan make:controller SettingController --resource

Setelah itu atur route dengan membuka file web.php kemudian tambahkan kode seperti dibawah ini.

```
Route::get('/setting','SettingController@index')->name('setting.transaksi')->middleware('role:admin');
```

Karena ada proses pengiriman data pada transaksi, maka route untuk form Setting Akun menggunakan **get**.

4. Membuat tampilan halaman data setting akun

Buat folder baru pada folder admin dengan nama **setting**, kemudian buat file pada file tersebut dengan cara klik kanan pada folder setting lalu pilih New File lalu berikan nama file **setting.blade.php** kemudian ketikan kode seperti dibawah ini

```
@extends('layouts.layout')  
@section('content')  
@include('sweetalert::alert')  
<div class="d-sm-flex align-items-center justify-content-between mb-4">  
<h1 class="h3 mb-0 text-gray-800">Setting Akun untuk transaksi </h1>  
</div>
```

```

<hr>
<form action="/setting/simpan" method="POST">
    @csrf
    @foreach ($setting as $stg)
        <div class="row col-sm-6">
            <div class="col-sm">
                <input type="hidden" name="kode[]" value="{{ $stg->id_setting }}">
                <label for="exampleFormControlInput1">Transaksi {{ $stg-
>nama_transaksi}} </label>
            </div>
            <div class="col-sm">
                <label for="exampleFormControlInput1">{{ $stg->no_akun }}</label>
            </div>
            <div class="col-sm">
                <select name="akun[]" id="supp_select2" class="form-
control" required width="100%">
                    <option value="">Pilih Akun</option>
                    @foreach ($akun as $akn)
                        <option value="{{ $akn->no_akun }}">{{ $akn->no_akun }} - {{ $akn-
>nm_akun }} </option>
                    @endforeach
                </select>
            </div>
        </div>
    @endforeach
</form>
@endsection

```

Untuk menampilkan data yang terdapat pada tabel akun pada halaman index akun, tambahkan kode untuk mengambil data pada file **SettingController.php**, pada action index tuliskan kode berikut ini.

```

public function index()
{
    $akun=\App\Akun::All();
    $setting=\App\Setting::All();
    return view('setting.setting' ,
        ['akun' => $akun,
        'setting'=> $setting]);
}

public function simpan(Request $request){
    $kode = $request->kode;
    $akun = $request->akun;
    foreach($akun as $key => $no)
    {
        $input['no_akun'] = $akun[$key];
    }
}

```

```

        Setting::where('id_setting',$kode[$key])->update($input);
    }
    Alert::warning('Pesan ','Setting Akun telah dilakukan ');
    return redirect('setting');
}

```

5. Input data setting akun menggunakan seeder

Untuk membuat seeder setting maka silahkan gunakan perintah dibawah ini pada terminal:

php artisan make:seeder SettingSeeder

apabila berhasil silahkan buka file **SettingSeeder.php** pada folder database/seed, kemudian masukan data record dengan perintah dibawah ini:

```

public function run()
{
    $setting = Setting :: create([
        'id_setting' => '1',
        'no_akun' => '501',
        'nama_transaksi' => 'Retur',
    ]);

    $setting = Setting :: create([
        'id_setting' => '2',
        'no_akun' => '500',
        'nama_transaksi' => 'Pembelian',
    ]);

    $setting = Setting :: create([
        'id_setting' => '3',
        'no_akun' => '101',
        'nama_transaksi' => 'Kas',
    ]);
}

```

Setelah code diatas selesai dikerjakan, maka silahkan file **DatabaseSeeder.php** dan masukan source code dibawah ini

```
$this->call(SettingSeeder::class);
```

Dan jalankan perintah seeder dengan perintah:

php artisan db:seed

setelah file seeder dijalankan, silahkan tambahkan route pada web.php untuk pemanggilan data setting akun

```

//Setting
Route::get('/setting','SettingController@index')->name('setting.transaksi')->middleware('role:admin');
Route::post('/setting/simpan','SettingController@simpan');

```

Beranda

Data Master >

Menu Utama 2 >

Menu Utama 3

Setting Akun untuk transaksi

Transaksi Retur	501	Pilih Akun ▾
Transaksi Pembelian	500	Pilih Akun ▾
Transaksi Kas	101	Pilih Akun ▾

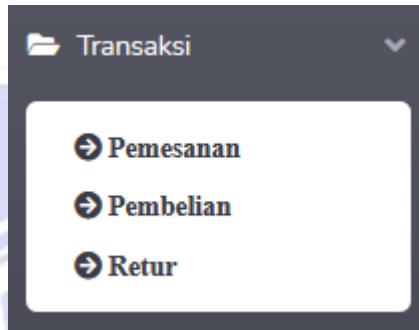
Gambar: Tampil Form Data Setting Akun



PERTEMUAN 9

9.1. Membuat Form Transaksi Pemesanan

1. Setting Struktur Navigasi Form Transaksi



Gambar: Menu Struktur Navigasi Form Transaksi

Buka file **layout.blade.php**, rubah **menu utama 2** menjadi **Transaksi**, kemudian **Sub menu 1** rubah menjadi **Pemesanan**, **Sub menu 2** rubah menjadi **Pembelian** dan **Sub menu 3** rubah menjadi **Retur**. Tampilan Sourcecode seperti dibawah ini:

```
<li class="nav-item">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapsePages1" aria-expanded="true" aria-controls="collapsePages1">
        <i class="fas fa-fw fa-folder-open"></i>
        <span>Transaksi</span>
    </a>
    <div id="collapsePages1" class="collapse" aria-
labelledby="headingPages" data-parent="#accordionSidebar">
        <div class="bg-white py-2 collapse-inner rounded">
            <a class="collapse-item fas fa-arrow-circle-
right" href="#">Pemesanan</a>
            <a class="collapse-item fas fa-arrow-circle-
right" href="#">Pembelian</a>
            <a class="collapse-item fas fa-arrow-circle-right" href="#">Retur</a>
        </div>
    </div>
</li>
```

2. Membuat Model Pemesanan

Langkah selanjutnya, tambahkan model **Pemesanan**, **Pemesanan_tem**, **Temp_pesanan.php** dan **Detail_Pesanan.php**, akan tetapi sudah dibuatkan di pertemuan pertama, maka silahkan buka model **Pemesanan.php** pada folder **app/ Pemesanan.php** tambahkan source code sebagai berikut:

```

<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Pemesanan extends Model
{
    //jika tidak di definisikan makan primary akan terdetek id
    protected $primaryKey = 'no_pesan';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "pemesanan";
    protected $fillable=['no_pesan','tgl_pesan','total','kd_supp'];
}

```

Setelah itu buka model **Pemesanan_Tem.php**, dan tambahkan source code sebagai berikut:

```

<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Pemesanan_tem extends Model
{
    protected $primaryKey = 'kd_brg';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "temp_pemesanan";
    protected $fillable=['kd_brg','qty_pesan'];
}

```

File **Pemesanan_Tem** fungsinya untuk menyimpan data sementara transaksi pemesanan.

Selanjutnya buka model **Temp_pesanan.php**, dan tambahkan source code sebagai berikut:

```

<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Temp_pesanan extends Model
{
    protected $primaryKey = 'kd_brg';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "view_temp_pesanan";
    protected $fillable=['kd_brg','nm_brg','harga','stok'];
}

```


File **Temp_pesanan.php** fungsinya untuk menampilkan data view kedalam tabel temporary.

Selanjutnya buka model **Detail_pesanan.php**, dan tambahkan source code sebagai berikut:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Temp_pesanan extends Model
{
    protected $primaryKey = 'no_pesanan';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "detail_pesanan";
    protected $fillable=['no_pesanan','kd_brg','qty_pesanan','subtotal'];
}
```

File **Detail_pesanan.php** fungsinya untuk menyimpan data pada table detail pesanan.

3. Membuat Controller Pemesanan

Buat controller resource baru dengan nama **PemesananController.php** dan **DetailPesananController.php**, dengan menggunakan terminal dengan perintah dibawah ini

```
php artisan make:controller PemesananController
dan
php artisan make:controller DetailPesananController
```

Setelah berhasil membuat Controller Pemesanan dan Detail Pemesanan, silahkan buka **PemesananController.php**, dan masukan source code dibawah ini;

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Barang;
use App\Supplier;
use App\Pemesanan;
use App\Pemesanan_tem;
use App\Temp_pesanan;
use Alert;
class PemesananController extends Controller
{
    public function index()
    {
        $akun=\App\Akun::All();
        $barang=\App\Barang::All();
        $supplier=\App\Supplier::All();
```

```

$temp_pesan=\App\Temp_pesan::All();
//No otomatis untuk transaksi pemesanan
$AWAL = 'TRX';
    $bulanRomawi = array("", "I","II","III", "IV", "V","VI","VII","VIII","IX","X"
, "XI","XII");
    $noUrutAkhir = \App\Pemesanan::max('no_pesan');
    $no = 1;
    $formatnya=sprintf("%03s", abs((int)$noUrutAkhir + 1)). '/' . $AWAL . '/' . $b
ulanRomawi[date('n')] . '/' . date('Y');
    return view('pemesanan.pemesanan' ,
        ['barang' => $barang,
        'akun' => $akun,
        'supplier'=>$supplier,
        'temp_pemesanan'=>$temp_pesan,
        'formatnya'=>$formatnya]);
}

public function tambahOrder()
{
    return view('pemesanan');
}
public function store(Request $request)
{
    //Validasi jika barang sudah ada paada tabel temporari maka QTY akan di edit
    if (Pemesanan_tem::where('kd_brg', $request->brg)->exists()) {
        Alert::warning('Pesan ', 'barang sudah ada.. QTY akan terupdate ?');
        Pemesanan_tem::where('kd_brg', $request->brg)
            ->update(['qty_pesan' => $request->qty]);
        return redirect('transaksi');
    }else{
        Pemesanan_tem::create([
            'qty_pesan' => $request->qty,
            'kd_brg' => $request->brg
        ]);
        return redirect('transaksi');
    }
}
public function destroy($kd_brg)
{
    //
    $barang=\App\Pemesanan_tem::findOrFail($kd_brg);
    $barang->delete();
    Alert::success('Pesan ', 'Data berhasil dihapus');
    return redirect('transaksi');
}

```

```
}  
}
```

Selanjutnya buka file **DetailPesanController.php**, dan masukan source code dibawa ini;

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
use App\DetailPesan;  
use App\Pemesanan;  
use Alert;
```

dan buatlah function simpan dan masukan source code dibawah ini:

```
public function simpan(Request $request)  
{  
    //Simpan ke table pemesanan  
    $tambah_pemesanan=new \App\Pemesanan;  
    $tambah_pemesanan->no_pesan = $request->no_pesan;  
    $tambah_pemesanan->tgl_pesan = $request->tgl;  
    $tambah_pemesanan->total = $request->total;  
    $tambah_pemesanan->kd_supp = $request->supp;  
    $tambah_pemesanan->save();  
    //Simpan data ke table detail pesan  
    $kd_brg = $request->kd_brg;  
    $qty= $request->qty_pesan;  
    $sub_total= $request->sub_total;  
    foreach($kd_brg as $key => $no)  
    {  
        $input['no_pesan'] = $request->no_pesan;  
        $input['kd_brg'] = $kd_brg[$key];  
        $input['qty_pesan'] = $qty[$key];  
        $input['subtotal'] = $sub_total[$key];  
        DetailPesan::insert($input);  
    }  
    Alert::success('Pesan ', 'Data berhasil disimpan');  
    return redirect('/transaksi');  
}
```

Controller Detail pesan ini fungsinya untuk menampung detail transaksi pemesanan.

Setelah itu atur route dengan membuka file web.php kemudian tambahkan kode seperti dibawah ini.

```
//Pemesanan  
Route::get('/transaksi', 'PemesananController@index')-  
>name('pemesanan.transaksi');  
Route::post('/sem/store', 'PemesananController@store');  
Route::get('/transaksi/hapus/{kd_brg}', 'PemesananController@destroy');
```

```
//Detail Pesan
Route::post('/detail/store', 'DetailPesanController@store');
Route::post('/detail/simpan', 'DetailPesanController@simpan');
```

4. Membuat tampilan transaksi pemesanan

Selanjutnya buat folder pemesanan dalam folder view kemudian buat file **pemesanan.blade.php**, dan ketikkan source code dibawah ini:

```
@extends('layouts.layout')
@section('content')
@include('sweetalert::alert')
<div class="d-sm-flex align-items-center justify-content-between mb-4">
<h1 class="h3 mb-0 text-gray-800">Transaksi Pemesanan </h1>
</div>
<hr>
<form action="/detail/simpan" method="POST">
    @csrf
    <div class="form-group col-sm-4">
        <label for="exampleFormControlInput1">No Faktur</label>

        <input type="text" name="no_pesan" value="{{ $formatnya }}" class="form-
control" id="exampleFormControlInput1" >
    </div>
    <div class="form-group col-sm-4">
        <label for="exampleFormControlInput1">Tanggal Transaksi</label>
        <input type="date" min="1" name="tgl" id="addnmbrg" class="form-
control" id="exampleFormControlInput1" require >
    </div>
    <div class="form-group col-sm-4">
        <label for="exampleFormControlInput1">Supplier</label>
        <select name="supp" id="supp select2" class="form-
control" required width="100%">
            <option value="">Pilih</option>
            @foreach ($supplier as $supp)
                <option value="{{ $supp->kd_supp }}">{{ $supp-
>nm_supp }} - {{ $supp->alamat }} </option>
            @endforeach
        </select>
    </div>
    <div class="card-header py-3 align="right">
    <button type="button" class="d-none d-sm-inline-block btn btn-sm btn-
danger shadow-sm" data-toggle="modal" data-target="#exampleModalScrollable">
    <i class="fas fa-plus fa-sm text-white-50"></i> Tambah Barang
    </button>
```

```

</div>
<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">
<div class="table-responsive">
<table class="table table-bordered table-
striped" id="dataTable" width="100%" cellpadding="0">
  <thead class="thead-dark">
    <tr>
      <th>Kode Barang</th>
      <th>Nama Barang</th>
      <th>Quantity</th>
      <th>Sub Total</th>
      <th>Aksi</th>
    </tr>
  </thead>
  <tbody>
    @php($total = 0)
    @foreach($temp_pemesanan as $temp)
      <tr>
        <td><input name="kd_brg[]" class="form-control" type="hidden" value="{{ $temp-
        >kd_brg }}" readonly >{{ $temp->kd_brg }}</td>
        <td><input name="nama" class="form-control" type="hidden" value="{{ $temp-
        >nm_brg }}" readonly >{{ $temp->nm_brg }}</td>
        <td><input name="qty_pesan[]" class="form-
        control" type="hidden" value="{{ $temp->qty_pesan }}" readonly>{{ $temp-
        >qty_pesan }}</td>
        <td><input name="sub_total[]" class="form-
        control" type="hidden" value="{{ $temp->sub_total }}" readonly >{{ $temp-
        >sub_total }}</td>
        <td align="center">
          <a href="/transaksi/hapus/{{ $temp-
          >kd_brg }}" onclick="return confirm('Yakin Ingin menghapus data?')" class="d-
          none d-sm-inline-block btn btn-sm btn-danger shadow-sm">
            <i class="fas fa-trash-alt fa-sm text-white-50"></i> Hapus</a>
          </td>
        </tr>
        @php($total += $temp->sub_total)
      @endforeach
      <tr>
        <td colspan="3"></td>
        <td><input name="total" class="form-
        control" type="hidden" value="{{ $total }}">Total {{ number_format($total) }}</a>
        <td></td>
      </tr>
    </tbody>
  </table>
</div>
</div>
</div>

```

```

</tr>
</tbody>
</table>
</div>
<input type="submit" class="btn btn-primary btn-send" value="Simpan Pemesanan">
</div>
</div>
</form>
<div class="modal fade" id="exampleModalScrollable" tabindex="-
1" role="dialog" aria-labelledby="exampleModalScrollableTitle" aria-
hidden="true">
<div class="modal-dialog modal-dialog-scrollable" role="document">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title" id="exampleModalScrollableTitle">Tambah Barang</h5>
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
<span aria-hidden="true">&times;</span>
</button>
</div>
<div>
<form action="/sem/store" method="POST">
@csrf
<div class="modal-body">
<div class="form-group">
<label for="exampleFormControlInput1">Barang</label>
<select name="brg" id="kd_brg select2" class="form-
control" required width="100%">
<option value="">Pilih</option>
@foreach ($barang as $product)
<option value="{{ $product->kd_brg }}">{{ $product-
>kd_brg }} - {{ $product->nm_brg }} [{{ number_format($product-
>harga) }}]</option>
@endforeach
</select>
</div>
<div class="form-group">
<label for="exampleFormControlInput1">QTY</label>
<input type="number" min="1" name="qty" id="addnmbrg" class="form-
control" id="exampleFormControlInput1" >
</div>
</div>
<div class="modal-footer">
<button type="button" class="btn btn-secondary" data-
dismiss="modal"> Batal</button>
<input type="submit" class="btn btn-primary btn-send" value="Tambah Barang">

```

```

</div>
</div>
</form>
</div>
</div>
@endsection

```

Perintah `@foreach` berfungsi untuk perulangan, pada form pemesanan ini terdapat dua `@foreach` yaitu untuk data supplier dan data barang.

Setelah itu rubah pemanggilan pada struktur navigasi yang terdapat pada file **layout.blade.php**. rubahlah source code dibawah ini:

```

<a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('pemesanan.transaksi') }}"> Pemesanan</a>

```

Silahkan coba jalankan form data pemesanan, berikut tampilan form pemesanan

Ketika tombol tambah barang diklik, maka akan muncul form data barang yang akan dibeli:

PERTEMUAN 10 & 11

10.1. Membuat Form Transaksi Pembelian

1. Membuat Model Pembelian

Buka model pembelian pada folder **app/Pembelian.php** tambahkan source code sebagai berikut:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Pembelian extends Model
{
    protected $primaryKey = 'no_beli';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "pembelian";
    protected $fillable=['no_beli','tgl_beli','no_faktur','total_beli','no_pesan'
];
}
```

Setelah itu buka model Detail pembelian pada folder **app/DetailPembelian.php** tambahkan source code sebagai berikut:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class DetailPembelian extends Model
{
    protected $primaryKey = 'no_beli';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "detail_pembelian";
    protected $fillable=['no_beli','kd_brg','qty_beli','sub_beli'];
}
```

Selanjutnya itu buka model **Beli.php** pada folder **app/Beli.php** tambahkan source code sebagai berikut:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Beli extends Model
{
```

```

protected $primaryKey = 'no_pesanan';
public $incrementing = false;
protected $keyType = 'string';
public $timestamps = false;
protected $table = "tampil_pemesanan";
protected $fillable=['kd_brg','no_pesanan','nm_brg','qty_pesanan','sub_total'];
}

```

2. Membuat Controller Pembelian

Buat controller resource baru dengan nama **PembelianController** dengan perintah dibawah ini:

php artisan make:controller PembelianController --resource

Setelah berhasil membuat Controller Pemesanan dan Detail Pemesanan, silahkan buka **PembelianController.php**, dan masukan source code dibawa ini:

```

<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Contracts\Encryption\DecryptException;
use Illuminate\Support\Facades\Crypt;
use App\DetailPembelian;
use App\Pembelian;
use DB;
use Alert;
use PDF;
class PembelianController extends Controller
{
    //
    public function index(){
        $pesan=\App\Pemesanan::All();
        //perintah SQL untuk menghilangkan data pembelian ketika sudah di beli
        $pesan = DB::select('SELECT * FROM pemesanan where not exists (select * from pembelian where pemesanan.no_pesanan=pembelian.no_pesanan)');
        //return view('pembelian.pembelian',['pemesanan'=>$pesan]);
    }

    public function edit($id){
        $AWAL = 'FKT';
        $bulanRomawi = array("", "I","II","III", "IV", "V","VI","VII","VIII","IX",
        ,"X", "XI","XII");
        $noUrutAkhir = \App\Pembelian::max('no_beli');
        $no = 1;
    }
}

```

```

        $format=sprintf("%03s", abs((int)$noUrutAkhir + 1)). '/' . $AWAL . '/' . $
bulanRomawi[date('n')] . '/' . date('Y');
        $AWALJurnal = 'JRU';
        $bulanRomawij = array("", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX",
        "X", "XI", "XII");
        $noUrutAkhirj = \App\Jurnal::max('no_jurnal');
        $noj = 1;
        $formatj=sprintf("%03s", abs((int)$noUrutAkhirj + 1)). '/' . $AWALJurnal
        . '/' . $bulanRomawij[date('n')] . '/' . date('Y');

        $decrypted = Crypt::decryptString($id);
        $detail      = DB::table('tampil_pemesanan')-
>where('no_pesanan',$decrypted)->get();
        $pemesanan  = DB::table('pemesanan')->where('no_pesanan',$decrypted)-
>get();
        $akunkas     = DB::table('setting')->where('nama_transaksi','Kas')-
>get();
        $akunpembelian = DB::table('setting')-
>where('nama_transaksi','Pembelian')->get();
        return view('pembelian.beli',['detail'=>$detail,'format'=>$format,'no_pes
an'=>$decrypted,'pemesanan'=>$pemesanan,'formatj'=>$formatj,'kas'=>$akunkas,'pemb
elian'=>$akunpembelian]);
    }

    public function pdf($id){
        $decrypted = Crypt::decryptString($id);
        $detail      = DB::table('tampil_pemesanan')-
>where('no_pesanan',$decrypted)->get();
        $supplier     = DB::table('supplier')->get();
        $pemesanan    = DB::table('pemesanan')->where('no_pesanan',$decrypted)-
>get();
        $pdf = PDF::loadView('laporan.faktur',['detail'=>$detail,'order'=>$pemesanan,
'supp'=>$supplier,'noorder'=>$decrypted]);
        return $pdf->stream();
    }

    public function simpan(Request $request)
    {
        if (Pembelian::where('no_pesanan', $request->no_pesanan)->exists()) {
            Alert::warning('Pesanan Pembelian Telah dilakukan ');

            return redirect('pembelian');
        }else{
            //Simpan ke table pembelian
            $tambah_pembelian=new \App\Pembelian;

```

```

$tambah_pembelian->no_beli = $request->no_faktur;
$tambah_pembelian->tgl_beli = $request->tgl;
$tambah_pembelian->no_faktur = $request->no_faktur;
$tambah_pembelian->total_beli = $request->total;
$tambah_pembelian->no_pesan = $request->no_pesan;
$tambah_pembelian->save();
//SIMPAN DATA KE TABEL DETAIL PEMBELIAN
$kdbrg = $request->kd_brg;
$qtybeli = $request->qty_beli;
$subbeli = $request->sub_beli;
foreach($kdbrg as $key => $no)
{
    $input['no_beli'] = $request->no_faktur;
    $input['kd_brg'] = $kdbrg[$key];
    $input['qty_beli'] = $qtybeli[$key];
    $input['sub_beli'] = $subbeli[$key];
    DetailPembelian::insert($input);
}
//SIMPAN ke table jurnal bagian debit
$tambah_jurnaldebit=new \App\Jurnal;
$tambah_jurnaldebit->no_jurnal = $request->no_jurnal;
$tambah_jurnaldebit->keterangan = 'Utang Dagang ';
$tambah_jurnaldebit->tgl_jurnal = $request->tgl;
$tambah_jurnaldebit->no_akun = $request->pembelian;
$tambah_jurnaldebit->debit = $request->total;
$tambah_jurnaldebit->kredit = '0';
$tambah_jurnaldebit->save();

//SIMPAN ke table jurnal bagian kredit
$tambah_jurnalkredit=new \App\Jurnal;
$tambah_jurnalkredit->no_jurnal = $request->no_jurnal;
$tambah_jurnalkredit->keterangan = 'Kas';
$tambah_jurnalkredit->tgl_jurnal = $request->tgl;
$tambah_jurnalkredit->no_akun = $request->akun;
$tambah_jurnalkredit->debit = '0';
$tambah_jurnalkredit->kredit = $request->total;
$tambah_jurnalkredit->save();
Alert::success('Pesan ', 'Data berhasil disimpan');
return redirect('/pembelian');
}
}

```

Setelah itu atur route dengan membuka file web.php kemudian tambahkan kode seperti dibawah ini.

```
//Pembelian
Route::get('/pembelian', 'PembelianController@index')-
>name('pembelian.transaksi');
Route::get('/pembelian-beli/{id}', 'PembelianController@edit');
Route::post('/pembelian/simpan', 'PembelianController@simpan');
```

Ada tiga route yang akan di buat yaitu untuk menampilkan, edit dan simpan.

3. Membuat Tampilan Transaksi Pembelian

Selanjutnya buat folder pembelian dalam folder view, kemudian buat file **pembelian.blade.php**, dan ketikkan source code dibawah ini:

```
@extends('layouts.layout')
@section('content')
@include('sweetalert::alert')
<div class="d-sm-flex align-items-center justify-content-between mb-4">
    <h1 class="h3 mb-0 text-gray-800">Transaksi Pembelian </h1>
</div>
<hr>
<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">

<div class="table-responsive">
<table class="table table-bordered table-
striped" id="dataTable" width="100%" cellpadding="0">
    <thead class="thead-dark">
        <tr>
            <th width="15%">No Pemesanan</th>
            <th>Tanggal Pesan</th>
            <th width="30%">Aksi</th>
        </tr>
    </thead>
    <tbody>
        @foreach($pemesanan as $pesan)
            <tr>
                <td width="15%">{{ $pesan->no_pesanan }}</td>
                <td>{{ $pesan->tgl_pesanan }}</td>
                <td width="30%">
                    <a href="{{url('/pembelian-beli/'.Crypt::encryptString($pesan-
>no_pesanan))}}" class="d-none d-sm-inline-block btn btn-sm btn-success shadow-
sm"><i class="fas fa-edit fa-sm text-white-50"></i> Beli</a>
                    <a href="{{route('cetak.order_pdf',[Crypt::encryptString($pesan-
>no_pesanan)])}}" target="_blank" class="d-none d-sm-inline-block btn btn-sm btn-
warning shadow-sm">
                        <i class="fas fa-print fa-sm text-white-50"></i> Cetak Invoice</a>
                </td>
            </tr>
        </tbody>
    </table>
</div>
</div>
</div>
```

```

</td>
</tr>
@endforeach
</tbody>
</table>
</div>
</div>
</div>
</form>
@endsection

```

Setelah itu membuat file baru pada folder pembelian dengan nama **beli.blade.php**, dan ketikkan source code dibawah ini:

```

@extends('layouts.layout')
@section('content')
@include('sweetalert::alert')
<div class="d-sm-flex align-items-center justify-content-between mb-4">

    <h1 class="h3 mb-0 text-gray-800">Pembelian </h1>
</div>
<hr>
<form action="/pembelian/simpan" method="POST">
    @csrf

    <div class="form-group col-sm-4">
        <label for="exampleFormControlInput1">No Pembelian</label>
        @foreach($kas as $ks)
            <input type="hidden" name="akun" value="{{ $ks-
>no_akun }}" class="form-control" id="exampleFormControlInput1" >
        @endforeach
        @foreach($pembelian as $bli)
            <input type="hidden" name="pembelian" value="{{ $bli-
>no_akun }}" class="form-control" id="exampleFormControlInput1" >
        @endforeach
        <input type="hidden" name="no_jurnal" value="{{ $formatj }}" class="f
orm-control" id="exampleFormControlInput1" >
        <input type="text" name="no_faktur" value="{{ $format }}" class="form
-control" id="exampleFormControlInput1" >
    </div>
    @foreach($pemesanan as $psn)
        <div class="form-group col-sm-4">
            <label for="exampleFormControlInput1">No Pemesanan</label>
            <input type="text" name="no_pesan" value="{{ $psn-
>no_pesan }}" class="form-control" id="exampleFormControlInput1" >
        </div>
    @endforeach
</form>

```

```

        <div class="form-group col-sm-4">
            <label for="exampleFormControlInput1">Tanggal Pemesanan</label>
            <input type="text" min="1" name="tgl" value="{{ $psn-
>tgl_pesan }}" id="addnmbrg" class="form-
control" id="exampleFormControlInput1" require >
        </div>
    @endforeach

<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">

<div class="table-responsive">
<table class="table table-bordered table-
striped" id="dataTable" width="100%" cellpadding="0">
    <thead class="thead-dark">
        <tr>
            <th>Kode Barang</th>
            <th>Nama Barang</th>
            <th>Quantity</th>
            <th>Sub Total</th>
            <th>Aksi</th>
        </tr>
    </thead>
    <tbody>
        @php($total = 0)
        @foreach($detail as $temp)
            <tr>
                <td><input name="no_beli[]" class="form-
control" type="hidden" value="{{ $temp-
>no_pesan }}" readonly><input name="kd_brg[]" class="form-
control" type="hidden" value="{{ $temp->kd_brg }}" readonly>{{ $temp->kd_brg }}</td>
                <td>{{ $temp->nm_brg }}</td>
                <td><input name="qty_beli[]" class="form-
control" type="hidden" value="{{ $temp->qty_pesan }}" readonly>{{ $temp-
>qty_pesan }}</td>
                <td><input name="sub_beli[]" class="form-
control" type="hidden" value="{{ $temp->sub_total }}" readonly>{{ $temp-
>sub_total }}</td>
                <td align="center">
                    <a href="/transaksi/hapus/{{ $temp-
>kd_brg }}" onclick="return confirm('Yakin Ingin menghapus data?')" class="d-
none d-sm-inline-block btn btn-sm btn-danger shadow-sm">
                    <i class="fas fa-trash-alt fa-sm text-white-50"></i> Hapus</a>

```



```

        </td>
    </tr>
    @php($total += $temp->sub_total)
    @endforeach
    <tr>
        <td colspan="3"></td>
        <td><input name="total" class="form-
control" type="hidden" value="{{ $total }}">Total {{ number_format($total) }}</a>
        <td ></td>
    </td>
    </tr>
</tbody>
</table>
</div>
<input type="submit" class="btn btn-primary btn-send" value="Simpan Pembelian">
</div>
</div>
</form>
@endsection

```

Setelah itu rubah pemanggilan pada struktur navigasi yang terdapat pada file **layout.blade.php**. rubahlah source code dibawah ini:

```

<a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('pembelian.transaksi') }}"> Pembelian</a>

```

Silahkan coba jalankan form transaksi pembelian, berikut tampilan form pembelian



Gambar: Tampilan Form Pembelian

Praktikum Pemrograman Akuntansi II

Pembelian

No Pembelian: 004FHT/V/2021

No Penyerahan: 005/TRA/V/2021

Tanggal Penyerahan: 2021-01-07

Kode Barang	Nama Barang	Quantity	Sub Total	Aksi
BR002	Rice Pile	3	400000	Hapus
BR003	Tasung Segitiga Biru	3	50000	Hapus
BR004	Tasung Cakra Kambur	1	20000	Hapus
			Total 520,000	

[Simpan Pembelian](#)

Gambar: Tampilan Form Detail Pembelian

9.2. Membuat Faktur Invoice

Untuk membuat Cetak Invoice silahkan buat folder baru pada views, dengan nama laporan, klik kanan klik new file, berikan nama **faktur.blade.php**, dan ketik source code dibawah ini:

```
<html>
<head>
<meta charset="utf-8">
<title>Invoice PT XYZ</title>
<style>
.invoice-box {
  max-width: 800px;
  margin: auto;
  padding: 30px;
  border: 1px solid #eee;
  box-shadow: 0 0 10px rgba(0, 0, 0, .15);
  font-size: 16px;
  line-height: 24px;
  font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
  color: #555;
}

.invoice-box table {
  width: 100%;
  line-height: normal; /* inherit */
  text-align: left;
}
```

```

.invoice-box table td {
    padding: 5px;
    vertical-align: top;
}
.invoice-box table tr td:nth-child(2) {
    text-align: right;
}
.invoice-box table tr.top table td {
    padding-bottom: 20px;
}
.invoice-box table tr.top table td.title {
    font-size: 45px;
    line-height: 45px;
    color: #333;
}
.invoice-box table tr.information table td {
    padding-bottom: 40px;
}
.invoice-box table tr.heading td {
    background: #eee;
    border-bottom: 1px solid #ddd;
    font-weight: bold;
}
.invoice-box table tr.details td {
    padding-bottom: 20px;
}
.invoice-box table tr.item td{
    border-bottom: 1px solid #eee;
}
.invoice-box table tr.item.last td {
    border-bottom: none;
}
.invoice-box table tr.total td:nth-child(2) {
    border-top: 2px solid #eee;
    font-weight: bold;
}
@media only screen and (max-width: 600px) {
    .invoice-box table tr.top table td {
        width: 100%;
        display: block;
        text-align: center;
    }
    .invoice-box table tr.information table td {
        width: 100%;
    }
}

```

```

        display: block;
        text-align: center;
    }
}
/** RTL **/
.rtl {
    direction: rtl;
    font-family: Tahoma, 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
}
.rtl table {
    text-align: right;
}
.rtl table tr td:nth-child(2) {
    text-align: left;
}
</style>
</head>
<body>
<div class="invoice-box">
    <table cellpadding="0" cellspacing="0">
        <tr class="top">
            <td colspan="2">
                <table>
                    <tr>
                        <td class="title">
                            
                        </td>
                        <td>
                            Invoice : <strong>#{{ $noorder }}</strong><br>
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr class="information">
            <td colspan="2">
                <table>
                    <tr>
                        <td>
                            @foreach($order as $pesan)
                            @foreach($supp as $sp)
                            @if($pesan->kd_supp==$sp->kd_supp)
                            <strong>PENERIMA</strong><br>

```

```

        {{ $sp->nm_supp }}<br>
        {{ $sp->alamat }}<br>
        {{ $sp->telepon }}<br>
    </td>
    @endif
    @endforeach
    @endforeach
    <td>
        <strong>PENGIRIM</strong><br>
        BSI Sukabumi<br>
        026689788<br>
        Jl Cemerlang No.8 <br>
        Sukakarya, Kota Sukabumi<br>
        Jawa Barat
    </td>
</tr>
</table>
</td>
</tr>
<tr class="heading">
    <td>Produk</td>
    <td>Subtotal</td>
</tr>
@php($total = 0)
@foreach ($detail as $row)
    <tr class="item">
        <td>
            {{ $row->nm_brg }}<br>
            <strong>Harga</strong>: Rp {{ number_format($row->sub_total/$row->qty_pesan) }} x {{ $row->qty_pesan }}
        </td>
        <td>Rp {{ number_format($row->sub_total) }}</td>
    </tr>
    @php($total += $row->sub_total)
@endforeach

    <tr class="total">
        <td></td>
        <td>
            Total: Rp {{ number_format($total) }}
        </td>
    </tr>
</tr>

```

```
 Mohon segera dikirim permintaan barang yang tertera d i atas</td> </tr> </table> </div> </body> </html> | | |
```

Dan berikut tampilan invoice:

Invoice : #005/TRX/I/2021

PENERIMA
cacah
sukabumi
0889

PENGIRIM
BSI Sukabumi
026689788
Jl Cemerlang No.8
Sukakarya, Kota Sukabumi
Jawa Barat

Produk	Subtotal
Apa Aja Harga: Rp 200,000 x 2	Rp 400,000
Tepung Segitiga Biru Harga: Rp 30,000 x 3	Rp 90,000
Tepung Cakra Kembar Harga: Rp 20,000 x 1	Rp 20,000
Total:	Rp 510,000

Mohon segera dikirim permintaan barang yang tertera diatas

Gambar: Tampilan Cetak Invoice

Setelah membuat code tampilan cetak invoice, silahkan tambahkan code untuk mengatur route dengan membuka file web.php kemudian tambahkan code seperti dibawah ini.

```

//Cetak Invoice
Route::get('/laporan/faktur/{invoice}', 'PembelianController@pdf')-
>name('cetak.order_pdf');

```

PERTEMUAN 12

12.1. Membuat Trigger Pengurangan Stok

Buatlah Trigger baru untuk update stok pengurangan dengan nama **trigger_kurang**, trigger ini berfungsi pada saat melakukan retur barang, maka stok barang otomatis akan berkurang, berikut perintah membuat trigger kurang:

php artisan make:migration trigger_kurang

Silahkan buka file **trigger_kurang**, dan masukan code dibawah ini:

```
public function up()
{
    DB::unprepared('
    CREATE TRIGGER update_stok_retur after INSERT ON detail_retur
    FOR EACH ROW BEGIN
    UPDATE barang
    SET stok = stok - NEW.qty_retur
    WHERE
    kd_brg = NEW.kd_brg;
    END
    ');
}
public function down()
{
    DB::unprepared('DROP TRIGGER update_stok_retur');
}
}
```

Selanjutnya lakukan migrate dengan perintah **php artisan migrate**.

12.2. Membuat Form Retur

1. Membuat Model Retur dan Detail Retur

Langkah selanjutnya, tambahkan model **Retur** dan **DetailRetur**, akan tetapi sudah dilakukan di pertemuan pertama, maka silahkan buka model **Retur.php** dan **DetailRetur.php** pada folder **app/Retur.php** tambahkan source code sebagai berikut

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Retur extends Model
{
    protected $primaryKey = 'no_retur';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "retur_beli";
    protected $fillable=['no_retur','tgl_retur','total_retur'];
}
```


dan buka model **DetailRetur.php**, tambahkan source code berikut ini:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class DetailRetur extends Model
{
    protected $primaryKey = 'no_retur';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "detail_retur";
    protected $fillable=['no_retur','kd_brg','qty_retur','sub_retur'];
}
```

2. Membuat Controller Retur

Buat controller resource baru dengan nama **ReturController.php**, dengan menggunakan terminal dengan perintah dibawah ini

php artisan make:controller ReturController --resource

Setelah berhasil membuat Controller Retur, silahkan buka **ReturController.php**, dan masukan source code dibawah ini;

```
<?php
namespace App\Http\Controllers;
use Illuminate\Contracts\Encryption\DecryptException;
use Illuminate\Support\Facades\Crypt;
use Illuminate\Http\Request;
use App\Retur;
use App\DetailRetur;
use DB;
use Alert;
class ReturController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }
}
```

```

        $pembelian=\App\Pembelian::All();
        return view('retur.retur',['pembelian'=>$pembelian]);
    }

    public function edit($id){
        $AWAL = 'RTR';
        $bulanRomawi = array("", "I","II","III", "IV", "V","VI","VII","VIII",
"IX","X", "XI","XII");
        $noUrutAkhir = \App\Retur::max('no_retur');
        $no = 1;
        $format=sprintf("%03s", abs((int)$noUrutAkhir + 1)). '/' . $AWAL . '/'
. $bulanRomawi[date('n')] . '/' . date('Y');
        //No otomatis untuk jurnal
        $AWALJurnal = 'JRU';
        $bulanRomawij = array("", "I","II","III", "IV", "V","VI","VII","VIII"
,"IX","X", "XI","XII");
        $noUrutAkhirj = \App\Jurnal::max('no_jurnal');
        $noj = 1;
        $formatj=sprintf("%03s", abs((int)$noUrutAkhirj + 1)). '/' . $AWALJur
nal . '/' . $bulanRomawij[date('n')] . '/' . date('Y');

        $decrypted = Crypt::decryptString($id);
        $detail      = DB::table('tampil_pembelian')-
>where('no_beli',$decrypted)->get();
        $pemesanan   = DB::table('pemesanan')->where('no_pesanan',$decrypted)-
>get();
        $akunkas      = DB::table('setting')->where('nama_transaksi','Kas')-
>get();
        $akunretur    = DB::table('setting')-
>where('nama_transaksi','Retur')->get();
        return view('retur.beli',['beli'=>$detail,'format'=>$format,'no_pesanan'
=>$decrypted,'pemesanan'=>$pemesanan,'formatj'=>$formatj,'kas'=>$akunkas,'retur'
=>$akunretur]);
    }

    public function simpan(Request $request)
    {
        if (Retur::where('no_retur', $request->no_retur)->exists()) {
            Alert::warning('Pesan ', 'Retur sudah dilakukan ');
            return redirect('retur');
        }else{
            //SIMPAN DATA KE TABEL DETAIL RETUR
            $kdbrg = $request->kd_brg;
            $qtyretur = $request->jml_retur;

```

```

    $harga    = $request->harga;
    $total=0;
    foreach($kdbrg as $key => $no)
    {
        $input['no_retur']    = $request->no_retur;
        $input['kd_brg']      = $kdbrg[$key];
        $input['qty_retur']   = $qtyretur[$key];
        $input['sub_retur']   = $harga[$key]*$qtyretur[$key];

        DetailRetur::insert($input);
        $total=$harga[$key]*$qtyretur[$key];
    }
    //Simpan ke table retur
    $tambah_pembelian=new \App\Retur;
    $tambah_pembelian->no_retur = $request->no_retur;
    $tambah_pembelian->tgl_retur = $request->tgl;
    $tambah_pembelian->total_retur = $total;
    $tambah_pembelian->save();
    //SIMPAN ke table jurnal bagian debit
    $tambah_jurnaldebit=new \App\Jurnal;
    $tambah_jurnaldebit->no_jurnal = $request->no_jurnal;
    $tambah_jurnaldebit->keterangan = 'Kas';
    $tambah_jurnaldebit->tgl_jurnal = $request->tgl;
    $tambah_jurnaldebit->no_akun = $request->kas;
    $tambah_jurnaldebit->debit = $total;
    $tambah_jurnaldebit->kredit = '0';
    $tambah_jurnaldebit->save();

    //SIMPAN ke table jurnal bagian kredit
    $tambah_jurnalkredit=new \App\Jurnal;
    $tambah_jurnalkredit->no_jurnal = $request->no_jurnal;
    $tambah_jurnalkredit->keterangan = 'Retur Pembelian';
    $tambah_jurnalkredit->tgl_jurnal = $request->tgl;
    $tambah_jurnalkredit->no_akun = $request->retur;
    $tambah_jurnalkredit->debit = '0';
    $tambah_jurnalkredit->kredit = $total;
    $tambah_jurnalkredit->save();
    Alert::success('Pesan ', 'Data berhasil disimpan');
    return redirect('/retur');
}
}
}

```

Setelah itu atur route dengan membuka file web.php kemudian tambahkan kode seperti dibawah ini.

```
//Retur
Route::get('/retur','ReturController@index')->name('retur.transaksi');
Route::get('/retur-beli/{id}', 'ReturController@edit');
Route::post('/retur/simpan', 'ReturController@simpan');
```

3. Membuat Tampilan Transaksi Retur

Selanjutnya buat folder Retur dalam folder view kemudian buat file **retur.blade.php**, dan ketikkan source code dibawah ini:

```
@extends('layouts.layout')
@section('content')
@include('sweetalert::alert')
<div class="d-sm-flex align-items-center justify-content-between mb-4">
    <h1 class="h3 mb-0 text-gray-800">Transaksi Retur </h1>
</div>
<hr>
<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">

<div class="table-responsive">
<table class="table table-bordered table-
striped" id="dataTable" width="100%" cellpadding="0">
    <thead class="thead-dark">
    <tr>
    <th width="15%">No Pemesanan</th>
    <th>Tanggal Pesan</th>
    <th>Total</th>
    <th>Aksi</th>
    </tr>
    </thead>
    <tbody>
    @foreach($pembelian as $beli)
    <tr>
        <td width="15%">{{ $beli->no_faktur }}</td>
        <td>{{ $beli->tgl_beli }}</td>
        <td>Rp. {{ number_format($beli->total_beli) }}</td>
        <td width="30%">
            <a href="{{url('/retur-beli/'.Crypt::encryptString($beli-
>no_faktur))}}" class="d-none d-sm-inline-block btn btn-sm btn-primary shadow-
sm"><i class="fas fa-recycle fa-sm text-white-50"></i> Retur</a>
        </td>
    </tr>
    @endforeach
```

```

        </tbody>
    </table>
</div>
</div>
</div>
</form>
@endsection

```

Setelah itu rubah pemanggilan pada struktur navigasi yang terdapat pada file **layout.blade.php**. rubahlah source code dibawah ini:

```

<a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('retur.transaksi') }}"> Retur</a>

```

Setelah merubah struktur navigasi retur, buat file baru dengan nama **beli.blade.php** pada folder Retur, source code ini berfungsi untuk menampilkan detail form retur dan ketik source code dibawah ini:

```

@extends('layouts.layout')
@section('content')
@include('sweetalert::alert')
<div class="d-sm-flex align-items-center justify-content-between mb-4">

    <h1 class="h3 mb-0 text-gray-800"> Retur Pembelian </h1>
</div>
<hr>
<form action="/retur/simpan" method="POST">
    @csrf

    <div class="form-group col-sm-4">
        <label for="exampleFormControlInput1">No Retur</label>
        @foreach($kas as $ks)
            <input type="hidden" name="kas" value="{{ $ks-
>no_akun }}" class="form-control" id="exampleFormControlInput1" >
        @endforeach
        @foreach($retur as $rtr)
            <input type="hidden" name="retur" value="{{ $rtr-
>no_akun }}" class="form-control" id="exampleFormControlInput1" >
        @endforeach
        <input type="hidden" name="no_jurnal" value="{{ $formatj }}" clas
s="form-control" id="exampleFormControlInput1" >
        <input type="text" name="no_retur" value="{{ $format }}" class="f
orm-control" id="exampleFormControlInput1" readonly >
    </div>
    <div class="form-group col-sm-4">
        <label for="exampleFormControlInput1">Tanggal Retur</label>
        <input type="date" min="1" name="tgl" id="addnmbrg" class="form-
control" id="exampleFormControlInput1" required/>
    </div>

```

```

<div class="d-sm-flex align-items-center justify-content-between mb-4">
<div class="card-body">
<div class="table-responsive">
<table class="table table-bordered table-
striped" id="dataTable" width="100%" cellpadding="0">
    <thead class="thead-dark">
    <tr>
    <th>Kode Barang</th>
    <th>Nama Barang</th>
    <th align="center">Jumlah Barang Dibeli</th>
    <th width=10%>Jumlah Retur</th>
    </tr>
    </thead>
    <tbody>
    @php($total = 0)
    @foreach($beli as $bli)
    <tr>
        <td><input name="kd_brg[]" class="form-
control" type="hidden" value="{{ $bli->kd_brg }}" readonly>{{ $bli->kd_brg }}</td>
        <td>{{ $bli->nm_brg }}</td>
        <td align="center">
            <input name="qty_beli[]" class="form-
control" type="hidden" value="{{ $bli->qty_beli }}" readonly>
            <input name="harga[]" class="form-
control" type="hidden" value="{{ $bli->harga }}" readonly>
            {{ $bli->qty_beli }}</td>
        <td width=10%><input name="jml_retur[]" class="form-
control" type="number" value="0"></td>

    </tr>
    @endforeach
    </tbody>
    </table>
</div>
<input type="submit" class="btn btn-primary btn-send" value="Simpan Retur">
</div>
</div>
</form>
@endsection

```

Setelah code disimpan silahkan jalankan, maka tampilan form retur akan tampil seperti dibawah ini:

PEMROGRAMAN AKUNTANSI II

Sistem Informasi Akuntansi PT. XYZ

Admin

Transaksi Retur

Show 10 entries Search:

No Pemesanan	Tanggal Pesan	Total	Aksi
001/RTA/W2022	2022-05-12	Rp. 300.000	Retur

Showing 1 to 1 of 1 entries Previous Next

Gambar : Form Transaksi Retur

Pada saat tombol retur diklik maka form detail tampil retur seperti dibawah ini:

PEMROGRAMAN AKUNTANSI II

Sistem Informasi Akuntansi PT. XYZ

Admin

Retur Pembelian

No Retur: 002/RTB/W2022

Tanggal Retur: dd/mm/yyyy

Kode Barang	Nama Barang	Jumlah Barang Diketik	Jumlah Retur
B001	Pensil	50	0
B002	penghapus	20	0

Simpan Retur

Gambar : Form Transaksi Detail Retur

12.3. Membuat Laporan Jurnal Umum Dan Stok Barang

1. Setting Struktur Navigasi Form Laporan



Gambar: Menu Struktur Navigasi Form Laporan

Buka file **layout.blade.php**, rubah **menu utama 3** menjadi **Laporan**, kemudian **Sub menu 1** rubah menjadi **Jurnal Umum**, **Sub menu 2** rubah menjadi **Stok Barang**. Tampilan Sourcecode seperti dibawah ini:


```

<li class="nav-item">
    <a class="nav-link collapsed" href="#" data-toggle="collapse" data-
target="#collapsePages2" aria-expanded="true" aria-controls="collapsePages2">
        <i class="fas fa-fw fa-folder-open"></i>
        <span>Laporan</span>
    </a>
    <div id="collapsePages2" class="collapse" aria-
labelledby="headingPages" data-parent="#accordionSidebar">
        <div class="bg-white py-2 collapse-inner rounded">
            <a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('laporan.index') }}" > Jurnal Umum</a>
            <a class="collapse-item fas fa-arrow-circle-
right" href="{{ route('stok.index') }}"> Stok Barang</a>

        </div>
    </div>
</li>

```

2. Membuat Model Jurnal dan Laporan

Langkah selanjutnya, tambahkan model **Jurnal** dan **Laporan**, akan tetapi sudah dilakukan di pertemuan pertama, maka silahkan buka model **Jurnal.php** pada folder **app/Jurnal.php** tambahkan source code sebagai berikut:

```

<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Jurnal extends Model
{
    //jikatidak di definisikan makan primary akan terdetek id
    protected $primaryKey = 'no_jurnal';
    public $incrementing = false;
    protected $keyType = 'string';
    public $timestamps = false;
    protected $table = "jurnal";
    protected $fillable=[ 'no_jurnal','keterangan','tgl_jurnal','no_akun','debit','
kredit'];
}

```

Setelah itu buka model **laporan.php**, dan tambahkan source code sebagai berikut:

```

<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Laporan extends Model
{
    protected $table = "jurnal";
    protected $fillable = ['no_jurnal','tgl_jurnal','no_akun','debit','kredit'];
}

```

3. Membuat Controller Laporan

Buat controller resource baru dengan nama **LaporanController.php**, dengan menggunakan terminal dengan perintah dibawah ini

php artisan make:controller LaporanController --resource

Setelah berhasil membuat Controller Laporan, silahkan buka

LaporanController.php, dan masukan source code dibawah ini:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Laporan;
use App\Lap_jurnal;
use PDF;
use DB;
class LaporanController extends Controller
{
    //
    public function index()
    {
        return view('laporan.laporan');
    }

    public function show(Request $request)
    {
        $periode=$request->get('periode');
        if($periode == 'All')
        {
            $bb = \App\Laporan::All();
            $akun=\App\Akun::All();
            $pdf = PDF::loadview('laporan.cetak',['laporan'=>$bb,'akun' =
> $akun])->setPaper('A4','landscape');
            return $pdf->stream();
        }elseif($periode == 'periode'){
            $tglawal=$request->get('tglawal');
            $tglakhir=$request->get('tglakhir');
            $akun=\App\Akun::All();
            $bb=DB::table('jurnal')
                ->whereBetween('tgl_jurnal', [$tglawal,$tglakhir])
                ->orderBy('tgl_jurnal','ASC')
                ->get();
            $pdf = PDF::loadview('laporan.cetak',['laporan'=>$bb,'akun' =
> $akun])->setPaper('A4','landscape');
            return $pdf->stream();
        }
    }
}
```

```
}
```

4. Membuat Tampilan Laporan

Selanjutnya buka folder laporan pada view, kemudian buat file **laporan.blade.php**, dan ketikkan source code dibawah ini:

```
@extends('layouts.layout')
@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-12">
            <div class="card">
                <div class="cardheader">Laporan Transaksi Jurnal</div>
                <div class="card-body">
                    <form action="/laporan/cetak" method="PUT" target="_blank">
                        @csrf
                        <fieldset>
                            <div class="form-group row">
                                <div class="col-md-4">
                                    <label for="klasifikasi">Periode Jurnal</label>
                                <div class="col-md-8">
                                    <input id="jenis" type="hidden" name="jenis" value="bukubesar" class="form-control">
                                    <select id="periode" name="periode" class="form-control">
                                        <option value="">--Pilih Periode Laporan--</option>
                                        <option value="All">Semua</option>
                                        <option value="periode">Per Periode</option>
                                    </select>
                                </div>
                            </div>
                            <div class="col-md-3">
                                <label for="no_hp">Tanggal Awal</label>
                                <input id="tglawal" type="date" name="tglawal" class="form-control">
                            </div>
                            <div class="col-md-3">
                                <label for="no_hp">Tanggal Akhir</label>
                                <input id="tglakhir" type="date" name="tglakhir" class="form-control">
                            </div>
                        </div>
                        <div class="col-md-10">
                            <input type="submit" class="btn btn-success btnsend" value="Cetak" >
                        </div>
                    </fieldset>
                </form>
            </div>
        </div>
    </div>
</div>
@endsection
```

Gambar: Laporan Form Jurnal Umum

Untuk cetak laporannya, silahkan buat file dengan nama **cetak.blade.php** pada folder laporan dan masukan source code dibawah ini:

```
<!DOCTYPE html>
<html>
<head>
  <title>Laporan Buku Besar</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/
css/bootstrap.min.css" integrity="sha384-
gg0YR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQU0hcWr7x9JvoRxT2MZw1T" crossorigin="an
onymous">
  <style type="text/css">
    table tr td,
    table tr th{
      font-size: 10pt;
    }
  </style>
</head>
<body>
  <table class="table table-bordered" width="100%" align="center">
    <tr align="center">
      <td><h2>Laporan Jurnal<br>PT. Penjualan</h2>
      <hr>
      </td>
    </tr>
  </table>
  <table class="table table-bordered" width="100%" align="center">
    <thead>
      <tr>
```

```

        <th width="5%">Tanggal Transaksi</th>
        <th width="15%">Nama Akun/Perkiraan</th>
        <th width="5%">Debet</th>
        <th width="5%">Kredit</th>
    </tr>
</thead>
<tbody>
@php
$subtotal1 = 0;
$subtotal2 = 0;
@endphp
@foreach ($laporan as $akn)
    <!-- <tr>
        <td colspan="5">{{ $akn->tgl_jurnal }}</td>
    </tr> -->
    @foreach ($laporan as $bb)
        <!-- pembuatan prulangan bersarang -->
        @if($loop->parent->first)
            <tr>

                <td>{{ $bb->tgl_jurnal }}</td>
                <td>{{ $bb->no_jurnal }} {{ $bb->keterangan }}</td>
                <td>{{ number_format($bb->debet) }}</td>
                <td>{{ number_format($bb->kredit) }}</td>

            </tr>
            <!-- hitung total debet dan kredit -->
            {{ $subtotal1 += $bb->debet }};
            {{ $subtotal2 += $bb->kredit }};
        @endif

    @endforeach
@endforeach
<tr>

    <td></td>
    <td></td>
    <td>Rp. {{ number_format($subtotal1) }}</td>
    <td>Rp. {{ number_format($subtotal2) }}</td>

</tr>

</tbody>
</table>
<div align="right">
<h6>Tanda Tangan</h6><br><br><h6>{{ Auth::user()->name }}</h6>

```

```
</div>
</body>
</html>
```

Setelah itu atur route dengan membuka file web.php kemudian tambahkan kode seperti dibawah ini.

```
//Laporan
Route::resource( '/laporan' , 'LaporanController');
Route::resource( '/stok' , 'LapStokController');
//laporan cetak
Route::get('/laporancetak/cetak_pdf', 'LaporanController@cetak_pdf');
```

Laporan Jurnal PT. Penjualan			
Tanggal Transaksi	Nama Akun/Perkiraan	Debet	Kredit
2022-05-27	001/JRU/V/2022 Kas	12.500	0
2022-05-27	001/JRU/V/2022 Retur Pembelian	0	12.500
		Rp. 12.500	Rp. 12.500
Tanda Tangan			
Admin			

Gambar: Cetak Laporan Jurnal Umum

12.2. Laporan Stok Barang

1. Membuat Model Laporan Stok Barang

Langkah selanjutnya, tambahkan model Laporan stok, dengan perintah dibawah ini:

php artisan make:model Laporanstok

setelah berhasil, silahkan buka model **Laporanstok.php** pada folder **app/** **Laporanstok.php** tambahkan source code sebagai berikut:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Laporanstok extends Model
{
    protected $table = "lap_stok";
    protected $fillable = ['kd_brg','nm_brg','harga','stok','beli','retur'];
}
```

2. Membuat Controller Laporan Stok Barang

Buat controller resource baru dengan nama **LapstokController.php**, dengan menggunakan terminal dengan perintah dibawah ini

php artisan make:controller LapstokController --resource

Setelah berhasil membuat Controller Laporan Stok, silahkan buka masukan source code dibawa ini:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Laporanstok;
use PDF;
use DB;
class LapStokController extends Controller
{
    public function index()
    {
        $data = Laporanstok::All();
        return view ('laporan.stok',['data'=>$data]);
    }

    public function show(Request $request)
    {
        $periode=$request->get('periode');
        if($periode == 'All')
        {
            $bb = \App\Laporanstok::All();
            $akun=\App\Akun::All();
            $pdf = PDF::loadview('laporan.print',['laporan'=>$bb,'akun' =
> $akun])->setPaper('A4','landscape');
            return $pdf->stream();
        }elseif($periode == 'periode'){
            $tglawal=$request->get('tglawal');
            $tglakhir=$request->get('tglakhir');
            $akun=\App\Akun::All();
            $bb=DB::table('barang')
                ->whereBetween('tgl_jurnal', [$tglawal,$tglakhir])
                ->orderBy('tgl_jurnal','ASC')
                ->get();
            $pdf = PDF::loadview('laporan.print',['laporan'=>$bb,'akun' =
> $akun])->setPaper('A4','landscape');
            return $pdf->stream();
        }
    }
}
```



```
}
```

3. Membuat tampilan laporan stok

Selanjutnya buka folder laporan dalam folder view kemudian buat file **stok.blade.php**, dan ketikkan source code dibawah ini:

```
@extends('layouts.layout')
@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-12">
            <div class="card">
                <div class="cardheader">Laporan Stok</div>
                <div class="table-responsive">
                    <table class="table table-striped table-hover table-
bordered dt-responsive nowrap" cellspacing="0" width="100%">
                        <thead>
                            <tr>

                                <th>Kode</th>
                                <th>Nama</th>
                                <th>Stok Awal</th>
                                <th>Beli</th>
                                <th>Retur</th>
                                <th>Stok Total (Stok+Beli-retur)</th>
                            </tr>
                        </thead>
                        <tbody>
                            <?php
                                foreach ($data as $item):
                                    ?>
                                    <tr>
                                        <td>{{ $item->kd_brg}}</td>
                                        <td>{{ $item->nm_brg}}</td>
                                        <td>{{ number_format($item-
>stok,0,',','.') }}</td>
                                        <td>{{ number_format($item-
>beli,0,',','.') }}</td>
                                        <td>{{ number_format($item-
>retur,0,',','.') }}</td>
                                        <td>{{ number_format(($item->stok+$item-
>beli)-$item->retur) }}</td>
                                    </tr>
                                <?php
                                    endforeach;
                            </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
```

```

        ?>
      </tbody>
    </table>
  </div>
</div>
</div>
</div>
</div>
</div>
@endsection

```

PEMROGRAMAN AKUNTANSI II

Sistem Informasi Akuntansi PT. XYZ

Laporan Stok

Kode	Nama	Stok Awal	Beli	Retur	Stok Total (Stok+Beli-retur)
B001	Pensil	54	150	15	189
B002	penghapus	25	60	15	70

Gambar: Laporan Stok Barang



PERTEMUAN 13

Review pembahasan materi



PERTEMUAN 14 & 15

Presentasi Project



Daftar Pustaka

1. Azamudin, Muhammad. Mukhlisin, Hafid. 2018. "Laravel: The PHP framework for web artisans". Jakarta: Kungfu Koding.
2. Junirianto, Eko. 2018. "Pemrograman Web Dengan Framework Laravel". Ponorogo: PenerbitWade
3. Supardi, Yuniar. Sulaeman. 2019. "Semua Bisa Menjadi Programmer Laravel Basic". Jakarta: Elex Media Komputindo.
4. Yudhanto, Yudho. Prasetyo, Helmi Adi. 2019. "Mudah Menguasai Framework Laravel". Jakarta: Elex Media Komputindo.
5. Akbar, Nuris. "Membuat Laporan PDF Dengan DOMPdf Pada Laravel". <https://belajarpHP.net/laporan-pdf-dompdf-laravel/> (diakses pada bulan November 2020)
6. Kiddy. 2017. "Tutorial Laravel 5.4-Templating". <https://medium.com/@kiddy.xyz/halo-semua-kali-ini-gue-mau-ngajarin-templating-pake-laravel-9e3510f8a5ef> (diakses pada bulan November 2020)
7. <https://ngodingdata.com/cara-membuat-trigger-mysql/>
8. Farhan, Andre." Tutorial Belajar MySQL: Pengertian VIEW dan Cara Penggunaan VIEW dalam MySQL". <https://www.duniailkom.com/tutorial-belajar-mysql-pengertian-view-dan-cara-penggunaan-view-dalam-mysql/> (diakses pada bulan Desember 2020)
9. Candra, Alfian. "Catatan Laravel :: SweetAlert 2". <https://alfinchandra4.medium.com/catatan-laravel-sweetalert-2-c286678e23d0> (diakses pada bulan Januari 2021)
10. Ilmu Coding. "Cara Install dan Menggunakan Vue Js di Laravel 7". <https://ilmucoding.com/vue-js-laravel/> (diakses pada bulan desember 2020)

