KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Nama : Muhammad Bagus Indrawan
Kelas : TI-2H/20
NIM : 2241720217

Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

JOBSHEET 06

Template Form (AdminLTE), Server Validation, Client Validation, CRUD

Admin LTE adalah salah satu template yang sering digunakan oleh web developer sebagai template backend pada proyek yang sering dikerjakan. Jadi admin LTE ini adalah sebuah dashboard Administrator dibuat menggunakan bootstrap yang merupakan framework css yang paling banyak digunakan.

Dalam pertemuang kali ini kita akan memahami tentang bagainama cara mengoperasikan form pada AdminLTE, Client & server validation, dan interaksi CRUD.

Sesuai dengan `Studi Kasus PWL.pdf`.
Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS.**

Project **PWL_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

Laravel menyediakan beberapa pendekatan berbeda untuk memvalidasi data masuk aplikasi. Cara paling umum adalah menggunakan *validate method* yang tersedia pada semua permintaan HTTP yang masuk. Namun, kami juga akan membahas pendekatan validasi lainnya.

Proses validasi client side atau *client side validation* penerjemahannya lebih dilakukan oleh web browser sebagai client-nya, dimana didalam web browser sudah terdapat penggunaan library yang mampu menerjemahkan perintah yang dijalankan dihalaman web menggunakan client side scripting.

Contoh dari client side scripting ini antara lain seperti :

- HTML
- XHTML
- CSS
- JavaScript
- XML
- jQuery

Proses validasi dari *server side* dilakukan di sisi server, setelah data dikirimkan oleh browser dan berhasil diterima oleh server, selanjutnya didalam server terdapat modul yang dilakukan untuk memvalidasi data, sebelum akhirnya proses diteruskan.

Proses validasi di sisi server side dilakukan dengan menggunakan bahasa pemrograman seperti PHP atau SQL dan lainnya.

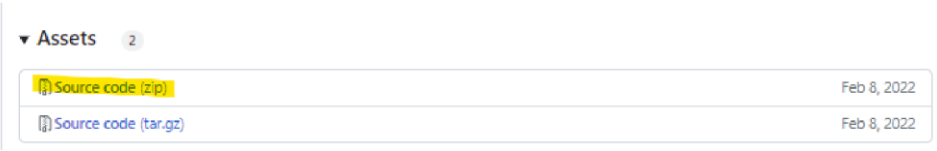
Contoh dari server side scripting antara lain:

- ASP (Active Server Pages)
- PHP Hypertext Processor
- JSP (Java Server Pages)
- Dll

Client side validation lebih dilakukan disisi browser dan bukan untuk tujuan keamanan, tetapi lebih ke kenyamanan pengguna. Sedangkan *server side validation* dilakukan di sisi server dengan tujuan keamanan dengan *filter* semua *request* yang masuk sebelum akhirnya diproses lanjutan.

A. Template Form (AdminLTE)

1. Penggunaan template AdminLTE Bootstrap Admin Dashboard Template dengan langkah pertama akses https://adminlte.io/ , lalu klik download pada source code (zip)



2. Lakukan extract All dan ubah nama folder *AdminLTE-3.2.0* menjadi **template,** folder template pindahkan ke laragon/www/PWL_POS/public. Buka folder public pada VSCode maka akan tampil folder baru bernama template

3. Copy isi public/template/index2.html pada welcome.blade.php lalukan edit agar dapat diload, edit href =”{{ asset (‘template/…..’) }}” dan src= ” {{ asset (‘template/…..’)” sesuai gambar berikut

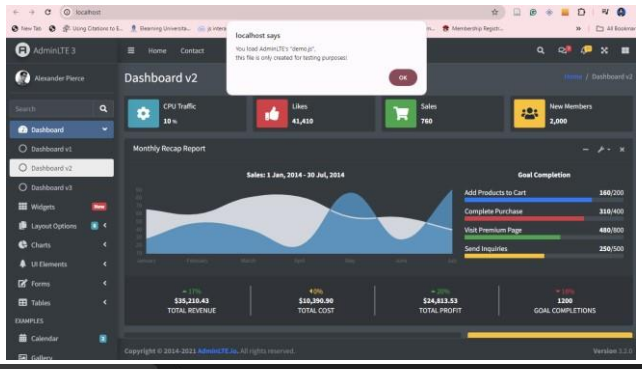
```
<!-- Google Font: Source Sans Pro -->
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
<!-- Font Awesome Icons -->
<link rel="stylesheet" href="{{ asset('template/plugins/fontawesome-free/css/all.min.css') }}">
<!-- overlayScrollbars -->
<link rel="stylesheet" href="{{ asset('template/plugins/overlayScrollbars/css/OverlayScrollbars.min.css') }}">
<!-- Theme style -->
<link rel="stylesheet"href="{{ asset('template/dist/css/adminlte.min.css') }}">
/head>
body class="hold-transition dark-mode sidebar-mini layout-fixed layout-navbar-fixed layout-footer-fixed">
div class="wrapper">

<!-- Preloader -->
<div class="preloader flex-column justify-content-center align-items-center">
  <img class="animation__wobble" src="{{asset('template/dist/img/AdminLTELogo.png') }}" alt="AdminLTELogo" height="60" width="60">
</div>
```

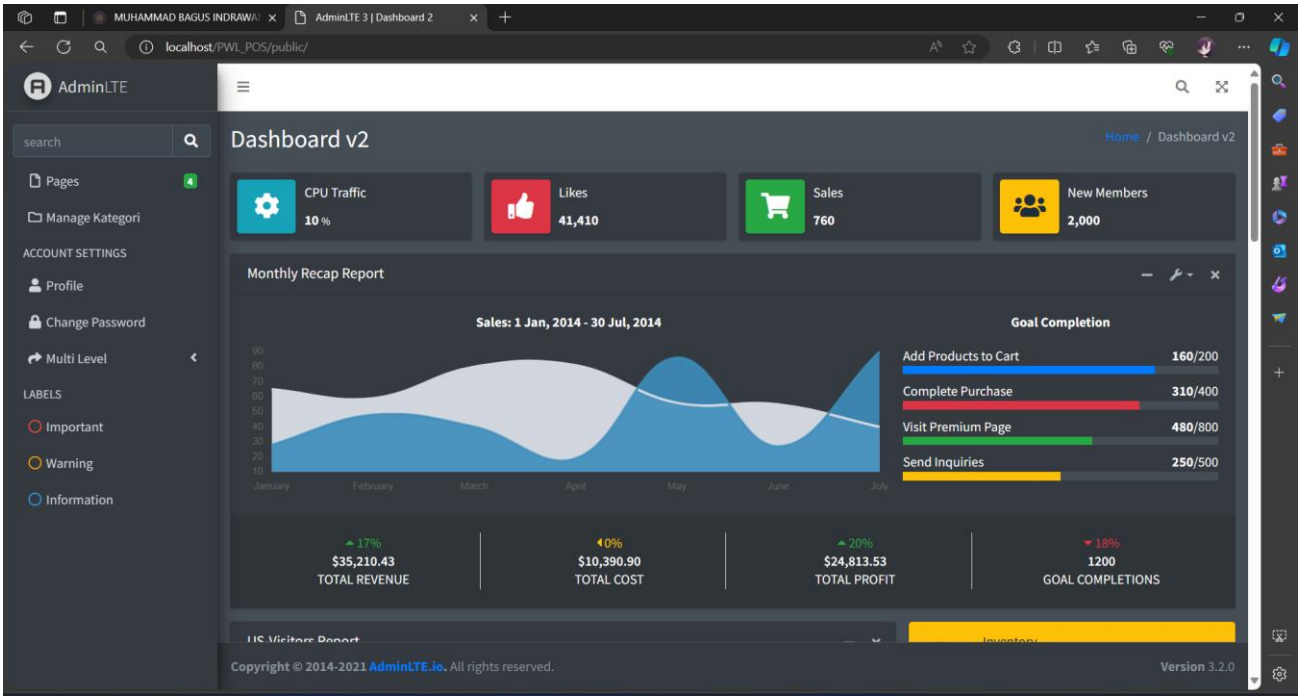
4. Scroll ke bagian paling bawah dan sesuaikan dengan kode berikut

```
1713  <!-- REQUIRED SCRIPTS -->
1714  <!-- jQuery -->
1715  <script src="{{ asset ('template/plugins/jquery/jquery.min.js') }}"></script>
1716  <!-- Bootstrap -->
1717  <script src="{{ asset ('template/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
1718  <!-- overlayScrollbars -->
1719  <script src="{{ asset ('template/plugins/overlayScrollbars/js/jquery.overlayScrollbars.min.js') }}"></script>
1720  <!-- AdminLTE App -->
1721  <script src="{{ asset ('template/dist/js/adminlte.js') }}"></script>
1722
1723  <!-- PAGE PLUGINS -->
1724  <!-- jQuery Mapael -->
1725  <script src="{{ asset ('template/plugins/jquery-mousewheel/jquery.mousewheel.js') }}"></script>
1726  <script src="{{ asset ('template/plugins/raphael/raphael.min.js') }}"></script>
1727  <script src="{{ asset ('template/plugins/jquery-mapael/jquery.mapael.min.js') }}"></script>
1728  <script src="{{ asset ('template/plugins/jquery-mapael/maps/usa_states.min.js') }}"></script>
1729  <!-- ChartJS -->
1730  <script src="{{ asset ('template/plugins/chart.js/Chart.min.js') }}"></script>
1731
1732  <!-- AdminLTE for demo purposes -->
1733  <script src="{{ asset ('template/dist/js/demo.js') }}"></script>
1734  <!-- AdminLTE dashboard demo (This is only for demo purposes) -->
1735  <script src="{{ asset ('template/dist/js/pages/dashboard2.js') }}"></script>
1736  </body>
1737  </html>
1738
```

Note: apabila gambar pada dist/img belum dapat diload maka lakukan penyesuaikan dengan asset `<img src="{{ asset('template/dist/img/user1-128x128.jpg')}}"`

5. Jalankan pada browser

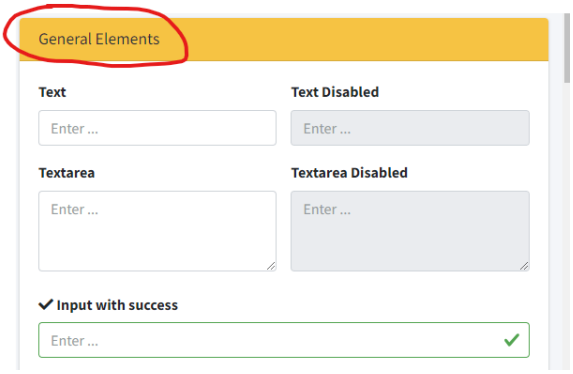




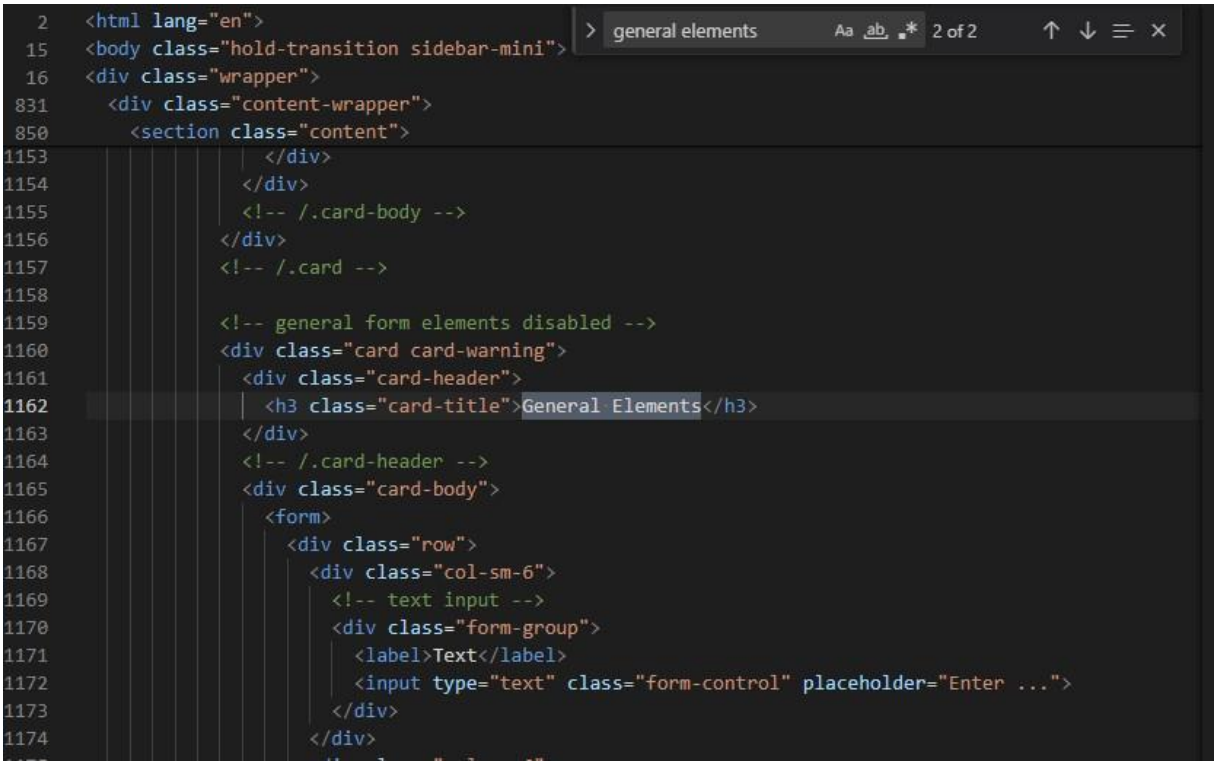
*note : template ini load untuk semuanya, hapus jika tidak diperlukan. Visual pada pada admin LTE dapat mempermudah untuk pengambilan kode.

6. Pada bagian menu terdapat pilihan Form kita khususkan dulu pada Form yang disediakan oleh Admin LTE
    - General Elements
    - Advanced Elements
    - Editor
    - Validation

    Cara menggunakan template bisa dilihat nama formnya contoh judulnya ‘General Elements’, agar dapat digunakan dapat melakukan akses pada folder yang terdapat template adminLTE contoh : C:\laragon\www\PWL_2024\public\template\pages\forms

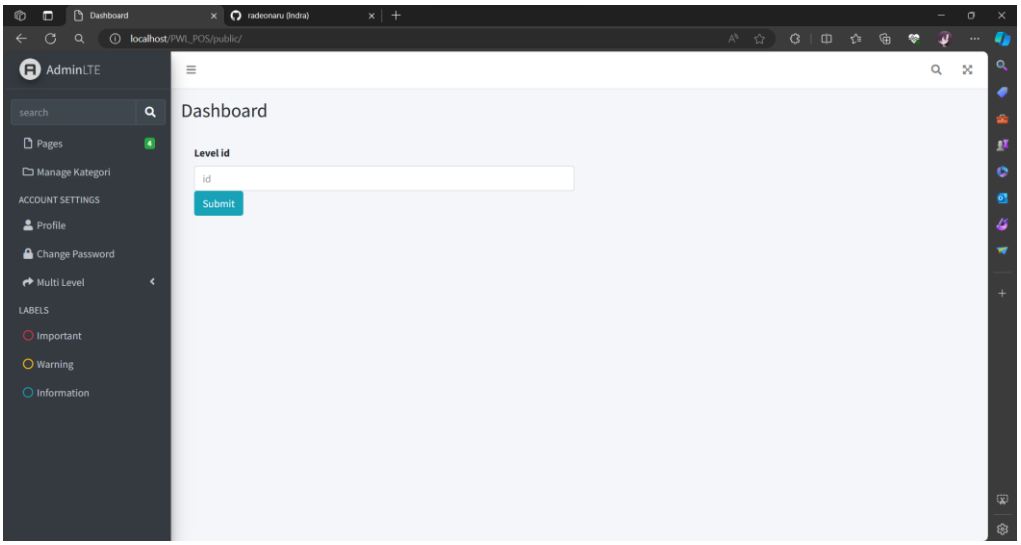


7. Buka file pada form> general cari keyword dengan ‘general elements’ dan coba modifikasi pada welcome.blade.php

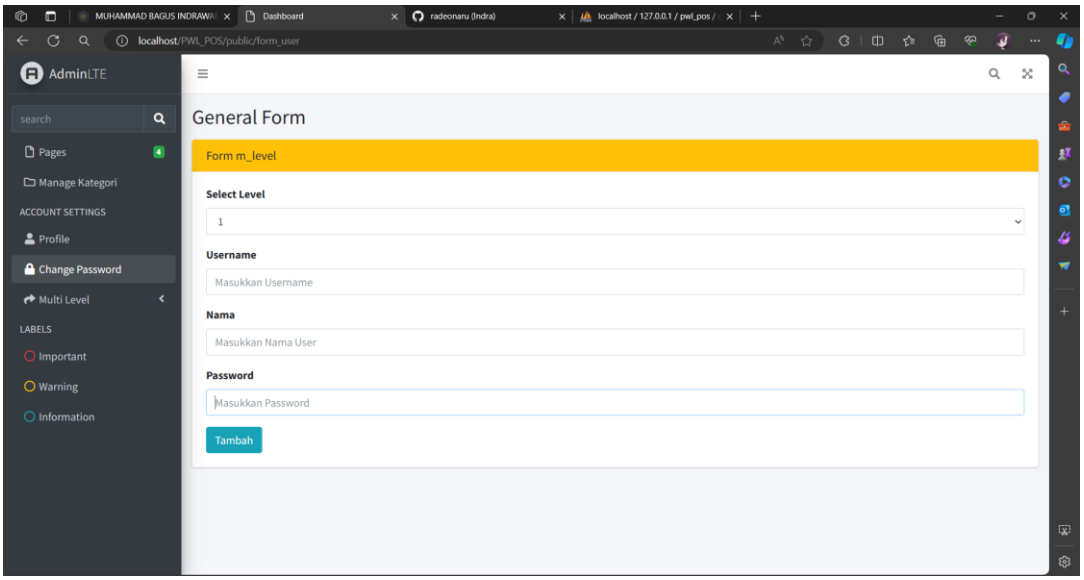


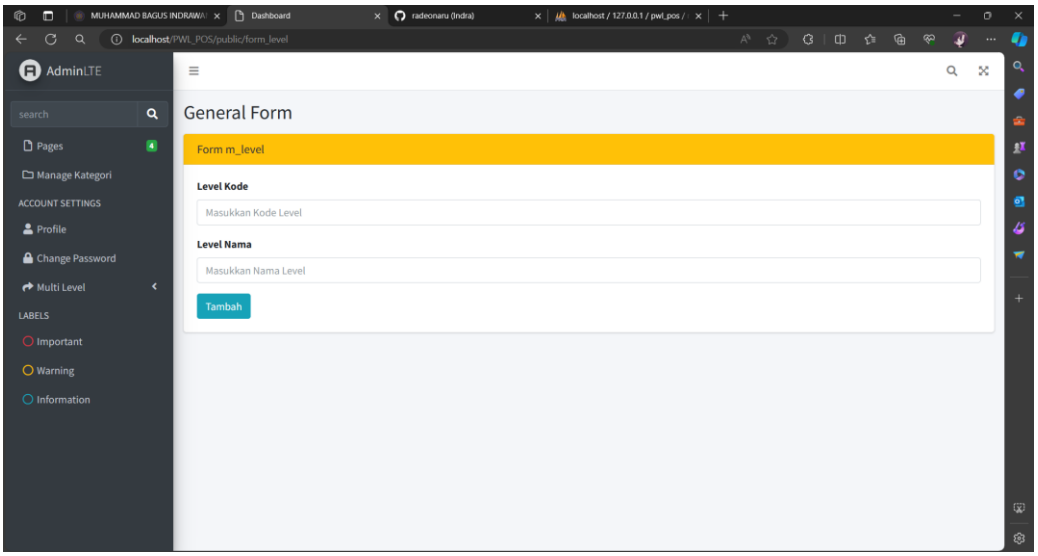
```
@extends('adminlte::page')

@section('title', 'Dashboard')

@section('content_header')
    <h1>Dashboard</h1>
@stop

@section('content')

    <div class="card-body">
        <form>
          <div class="row">
            <div class="col-sm-6">
              <!-- text input -->
              <div class="form-group">
     <label>Level id</label><input type="text" class="form-control" placeholder="id">
                <div>
</div>
              <button type = "submit" class ="btn btn-info">Submit </button>
            </div>
@stop

@section('css')
    {{-- Add here extra stylesheets --}}
    {{-- <link rel="stylesheet" href="/css/admin_custom.css"> --}}
@stop

@section('js')
    <script> console.log("Hi, I'm using the Laravel-AdminLTE package!"); </script>
@stop
```

8. Apa yang tampil?



9. Eksplorasi jenis form pada adminLTE dan coba terapkan yang sesuai untuk studi kasus POS PWL, buatlah form untuk tabel m_user dan m_level





B. VALIDASI PADA SERVER

1. Buka kembali file tugas di jobsheet 05, klik dan baca documentation ini Validation - Laravel 10.x - The PHP Framework For Web Artisans 2. Pastikan sudah terdefinisikan route pada web.php :

```
Route::get('/kategori/create', [KategoriController::class, 'create']);
Route::post('/kategori', [KategoriController::class, 'store']);
```

```
33
34  Route::get('/kategori/create', [KategoriController::class, 'create']);
35  Route::post('/kategori', [KategoriController::class. 'store']);
36
```

3. Edit pada KategoriKontroller dengan kode:

app > Http > Controllers > KategoriController.php > PHP > App\Http\Controllers\KategoriController

```php
namespace App\Http\Controllers;

use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\View\View;

3 references | 0 implementations
class KategoriController extends Controller
{
    /**
     * Show the form to create a new post.
     */
    0 references | 0 overrides
    public function create(): View
    {
        return view('kategori.create');
    }
    /**
     * Store a new post.
     */
    0 references | 0 overrides
    public function store(Request $request): RedirectResponse
    {
    $validated = $request->validate([
        'kategori_kode' => 'required',
        'kategori_nama' => 'required',
    ]);

    // The post is valid...

    return redirect('/kategori');
    }
}
```

4. Berikut ini sebagai alternatif contoh penulisan validasi dan bisa Anda sesuain katakata dengan apa yang sedang Anda kerjakan,

Alternatively, validation rules may be specified as arrays of rules instead of a single | delimited string:

```php
$validatedData = $request->validate([
    'title' => ['required', 'unique:posts', 'max:255'],
    'body' => ['required'],
]);
```

In addition, you may use the `validateWithBag` method to validate a request and store any error messages within a named error bag:

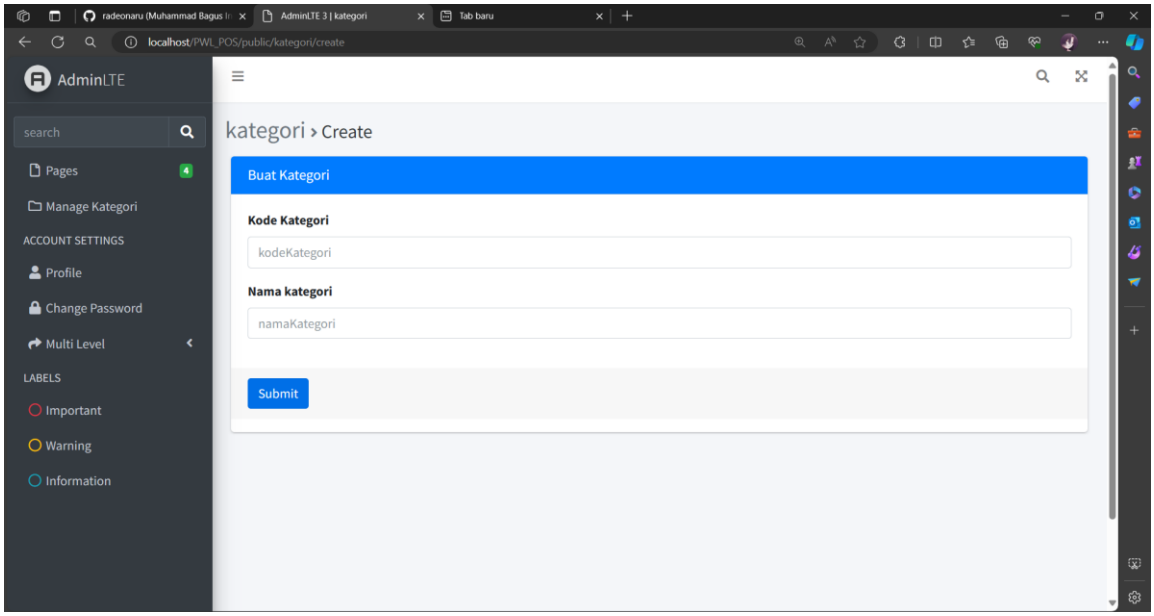
```php
$validatedData = $request->validateWithBag('post', [
    'title' => ['required', 'unique:posts', 'max:255'],
    'body' => ['required'],
]);
```

5. Tulis perbedaan penggunaan validate dengan validateWithBag!

Perbedaan validate dan validateWithBag adalah, hasil validasi validateWithBag akan disimpan dalam "bag" yang dapat disesuaikan (misalnya, "email"), sedangkan Hasil validasi validate akan disimpan dalam Session secara default.

6. Terkadang Anda mungkin ingin berhenti menjalankan aturan validasi pada suatu atribut setelah kegagalan validasi pertama. Untuk melakukannya, tetapkan **bail** aturan ke atribut: coba sesuaikan dengan field pada m_kategori, yuk bisa.. apa yang terjadi?

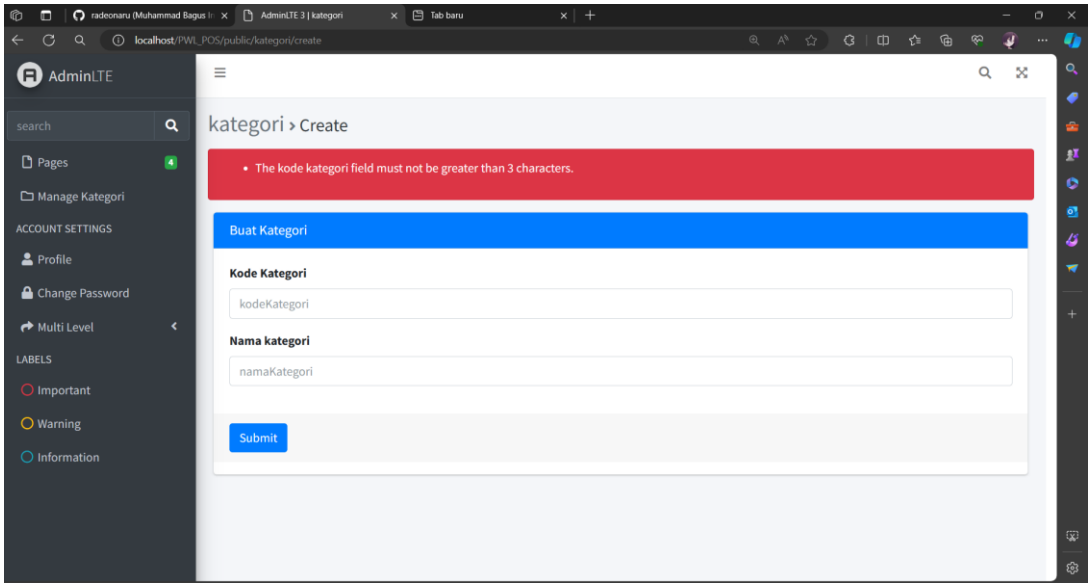
```
$request->validate([
    'title' => 'bail|required|unique:posts|max:255',
    'body' => 'required',
]);
```



7. Pada view/create.blade.php tambahkan code berikut agar ketika validasi gagal, kita dapat menampilkan pesan kesalahan dalam tampilan:

```
@if ($errors->any())
    <div class="alert alert-danger">
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif
```

8. Screenshot hasilnya.

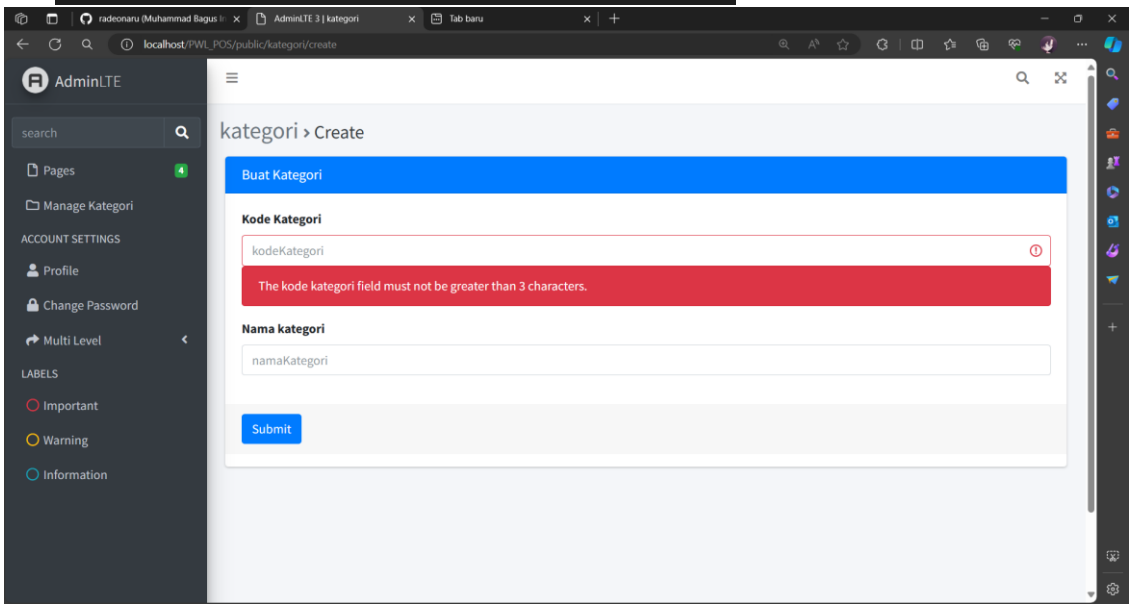


9. Catatan : Menyesuaikan Pesan Kesalahan

   Aturan validasi bawaan Laravel masing-masing memiliki pesan kesalahan yang terletak di lang/en/validation.php file aplikasi Anda. Jika aplikasi Anda tidak memiliki langdirektori, Anda dapat menginstruksikan Laravel untuk membuatnya menggunakan lang:publish perintah Artisan. Di dalam lang/en/validation.php file tersebut, Anda akan menemukan entri terjemahan untuk setiap aturan validasi. Anda bebas mengubah atau memodifikasi pesan-pesan ini berdasarkan kebutuhan aplikasi Anda.

10. Pada view/create.blade.php tambahkan dan coba running code berikut :

```
<label for="kategori_kode">Kode kategori</label>

<input id="kategori_kode"
    type="text"
    name="kategori_kode"
    class="@error('kategori_kode') is-invalid @enderror">

@error('kategori_kode')
    <div class="alert alert-danger">{{ $message }}</div>
@enderror
```



C. FORM REQUEST VALIDATION

1. Membuat permintaan form dengan menuliskan pada terminal

```
php artisan make:request StorePostRequest
```

kelas permintaan formulir yang dihasilkan akan ditempatkan di app/Http/Requestsdirektori. Jika direktori ini tidak ada, maka akan dibuat saat Anda menjalankan make:request perintah. Setiap permintaan formulir yang dihasilkan oleh Laravel memiliki dua metode: authorize dan rules.

```
PS C:\laragon\www\PWL_POS> php artisan make:request StorePostRequest

  INFO  Request [C:\laragon\www\PWL_POS\app\Http\Requests\StorePostRequest.php]
created successfully.

PS C:\laragon\www\PWL_POS>
```

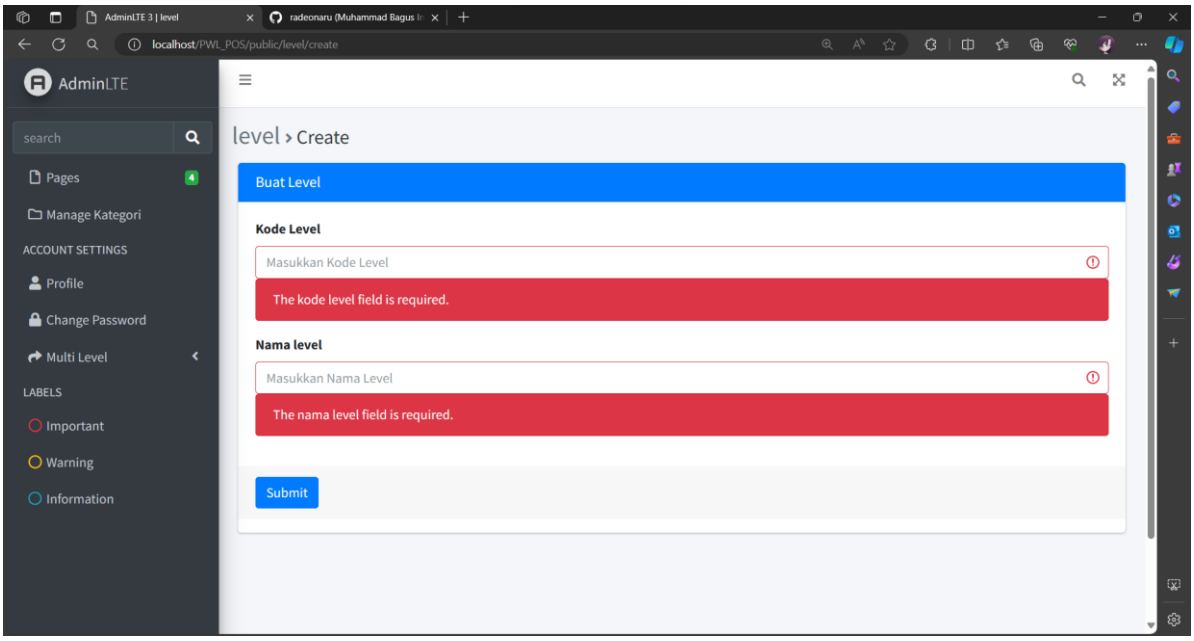
2. Ketik kode berikut pada Http/request/StorePostRequest

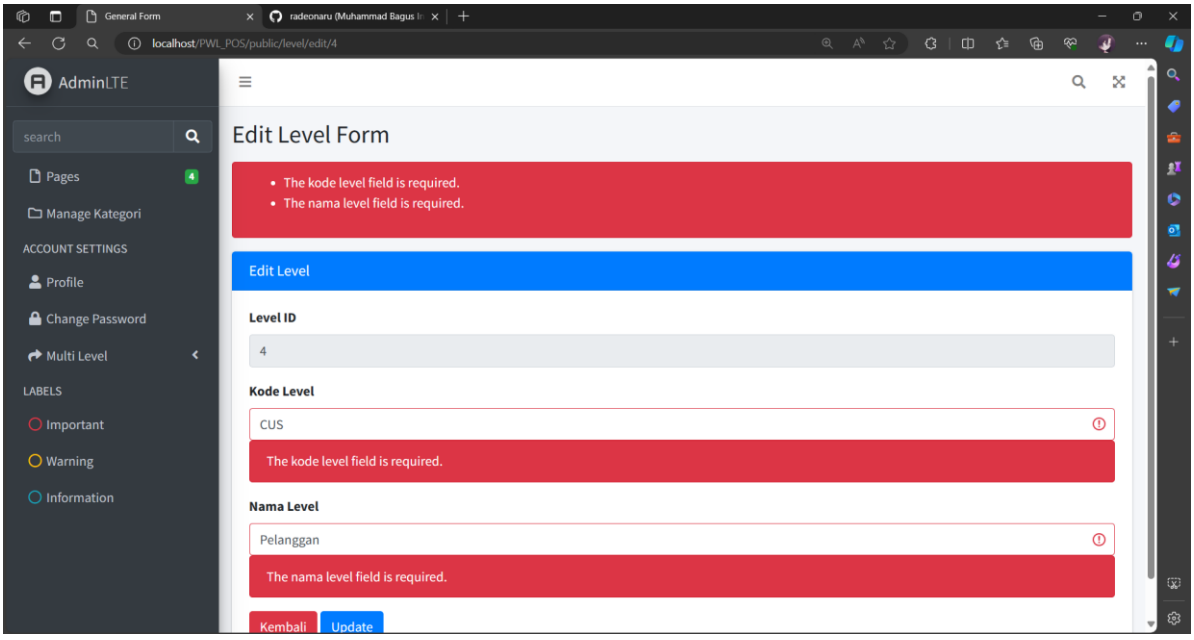
```php
/**
 * Get the validation rules that apply to the request.
 *
 * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
 */
0 references | 0 overrides
public function rules(): array
{
    return [
    'kategori_kode' => 'required',
    'kategori_nama' => 'required',
    ];
}
```

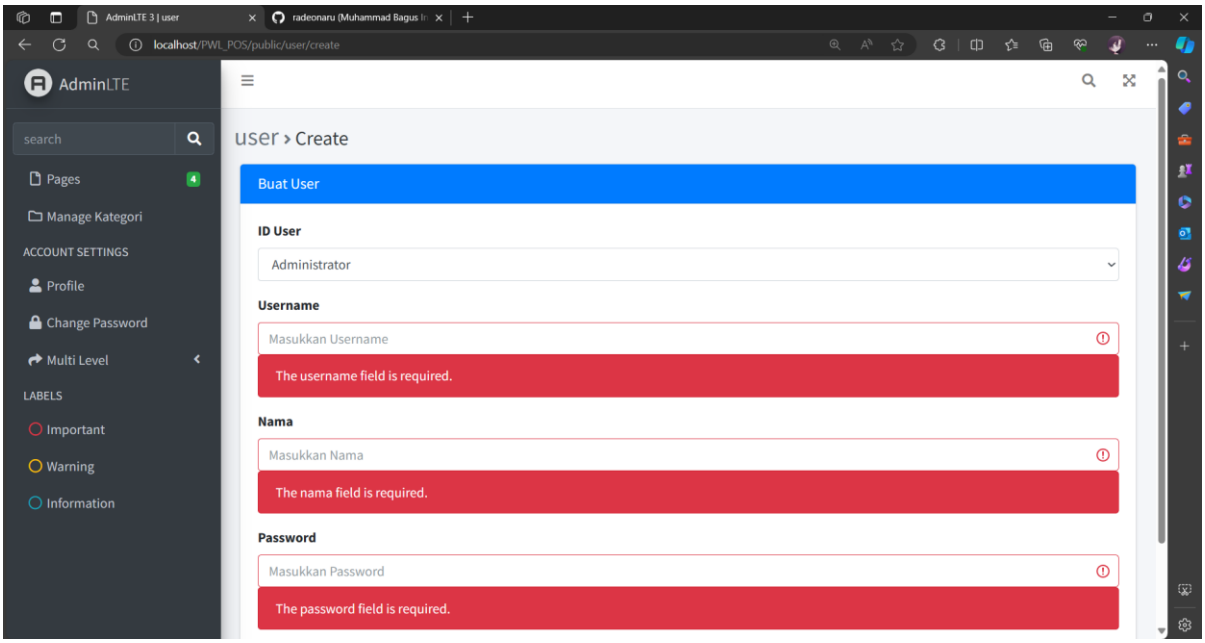
```php
0 references | 0 overrides
public function store(StorePostRequest $request): RedirectResponse
{
    // The incoming request is valid...

    // Retrieve the validated input data...
    $validated = $request->validated();

    // Retrieve a portion of the validated input data...
    $validated = $request->safe()->only(['kategori_kode', 'kategori_nama']);
    $validated = $request->safe()->except(['kategori_kode', 'kategori_nama']);

    // Store the post...

    return redirect ('/kategori');
}
```

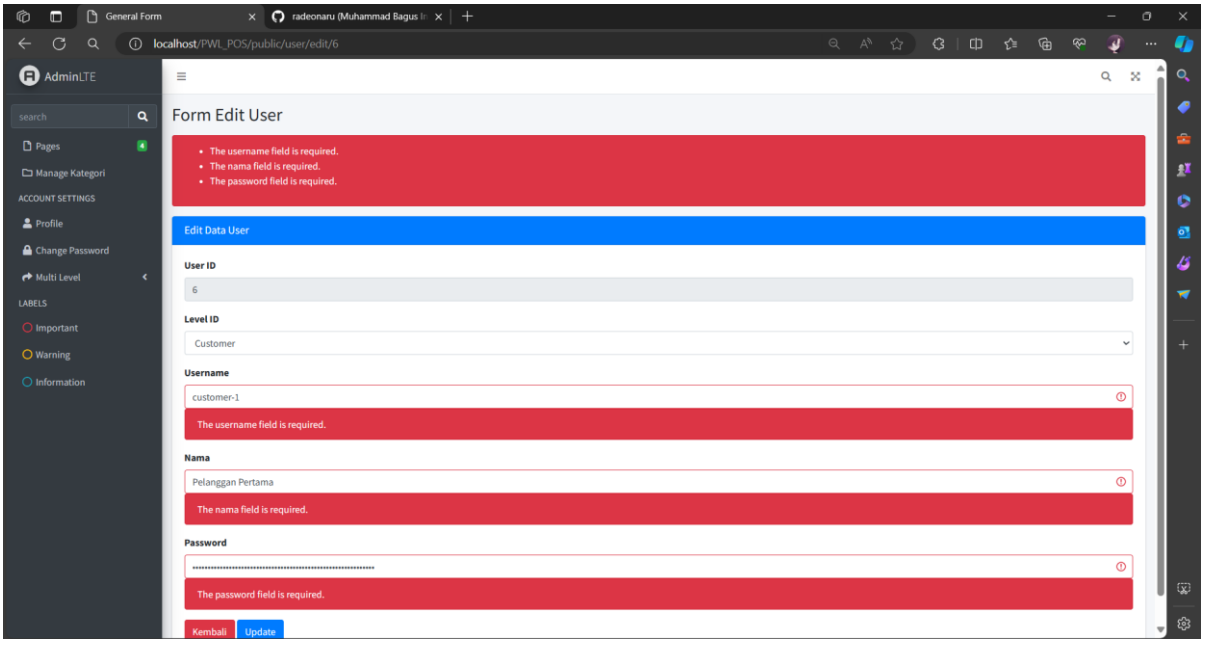
Note : Jika validasi gagal, respons pengalihan akan dihasilkan untuk mengirim pengguna kembali ke lokasi sebelumnya. Kesalahan juga akan ditampilkan ke sesi sehingga tersedia untuk ditampilkan. Jika permintaannya adalah permintaan XHR, respons HTTP dengan kode status 422 akan dikembalikan ke pengguna.

3. Terapkan validasi juga pada tabel m_user dan m_level.


AdminLTE 3 | level
localhost/PWL_POS/public/level/create
AdminLTE
search
Pages
Manage Kategori
ACCOUNT SETTINGS
Profile
Change Password
Multi Level
LABELS
Important
Warning
Information
level › Create
Buat Level
Kode Level
Masukkan Kode Level
The kode level field is required.
Nama level
Masukkan Nama Level
The nama level field is required.
Submit


General Form
localhost/PWL_POS/public/level/edit/4
AdminLTE
search
Pages
Manage Kategori
ACCOUNT SETTINGS
Profile
Change Password
Multi Level
LABELS
Important
Warning
Information
Edit Level Form
The kode level field is required.
The nama level field is required.
Edit Level
Level ID
4
Kode Level
CUS
The kode level field is required.
Nama Level
Pelanggan
The nama level field is required.
Kembali
Update





D. CRUD(create, read, update, delete)

1. Buat POSController lengkap dengan resourcenya, Membuat Resource Controller dan Route yang berfungsi untuk route CRUD sehingga tidak perlu repot-repot membuat masing-masing route seperti post, get, delete dan update.

```
php artisan make:controller POSController --resource
```

```
PS C:\laragon\www\PWL_POS2> php artisan make:controller POSController --resource

  INFO  Controller [C:\laragon\www\PWL_POS2\app\Http\Controllers\POSController.php] created successfully.

PS C:\laragon\www\PWL_POS2>
```

2. Berikut perintahnya: sekarang tambahkan kode pada route/web.php

```
Route::resource('m_user', POSController::class);
```

3. Atur pada Models/m_user

```
<
?
p
h
p
n
a
m
e
s
p
a
c
e
A
p
p
\
M
o
d
e
l
s
;
 use
Illuminate\Database\Eloquent\
Factories\HasFactory; use
Illuminate\Database\Eloquent\
Model;

c
l
a
```

```
s
s
m
_
u
s
e
r
e
x
t
e
n
d
s
M
o
d
e
l
{
protected
$table
="m_user";
public
$timestamps =
false;
protected
$primaryKey =
'user_id';

p
r
o
t
e
c
t
e
d
$
f
i
```

```
l
l
a
b
l
e
=
[
    'user_id',
    'level_id',

'
u
s
e
r
n
a
m
e
'
,
'
n
a
m
a
'
,
    'password',
   ];

}
```

4. Atur pada Migration/m_user_table

```
<?php  use App\Models\m_level; use
Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
 return new class extends
Migration
{
    /**
     * Run the migrations.
     */     public function
up(): void
    {
        Schema::create('useri', function (Blueprint $table) {
            $table-> id('user_id');
            $table -> unsignedBigInteger('level_id')->index;
            $table ->string ('username', 20)->unique();
            $table ->string ('nama', 100);
            $table ->string ('password');
            $table ->timestamps();

            $table->foreign('level_id')-> references('level_id')->on
('m_level');
        });
    }

    /**
     * Reverse the migrations.
     */     public function
down(): void
    {
        Schema::dropIfExists('useri');
    }
};
```

5. Pada file Controller yang kita dapat secara otomatis akan terdapat 7 fungsi berikut yang dapat kita gunakan untuk membuat operasi CRUD.
   - index()
   - create()
   - store()
   - show()
   - edit()

- update()
- destroy()

6. Selanjutnya bukalah app/Http/Controllers/POSController.php kemudian ketikkan, kodenya seperti berikut ini:

```php
<?php  namespace
App\Http\Controllers;
 use App\Models\m_user;
use
Illuminate\Http\Request;
 class POSController extends
Controller
{
    /**
* Display a listing of the resource.
     */     public
function index()
    {
        //fungsi eloquent menampilkan data menggunakan pagination
$useri = m_user::all(); // Mengambil semua isi tabel
return view('m_user.index', compact('useri'))->with('i');     }


    /**
* Show the form for creating a new resource.
     */     public
function create()
    {         return
view('m_user.create');
    }

    /**
* Store a newly created resource in storage.
     */     public function store(Request
$request)
    {
        //melakukan validasi data
     $request->validate([
    'user_id' => 'max 20',
    'username' => 'required',
    'nama' => 'required',

    ]);
    //fungsi eloquent untuk menambah data
```

```
    m_user::create($request->all());
     return redirect()->route('m_user.index')
    ->with('success', 'user Berhasil Ditambahkan');
    }

    /**
* Display the specified resource.
     */     public function show(string $id, m_user
$useri)
    {
        $useri = m_user::findOrFail($id);          return
view('m_user.show', compact('useri'));
    }

    /**
* Show the form for editing the specified resource.
     */     public function edit(string
$id)
    {
        $useri  =  m_user::find($id);              return
view('m_user.edit', compact('useri'));
    }

    /**
* Update the specified resource in storage.
     */     public function update(Request $request, string
$id)
    {
        $request->validate([

            'username'      =>      'required',
'nama' => 'required',
            'password' => 'required',
            ]);
            //fungsi eloquent untuk mengupdate data inputan kita
m_user::find($id)->update($request->all());
//jika data berhasil diupdate, akan kembali ke halaman utama return
redirect()->route('m_user.index')
->with('success', 'Data Berhasil Diupdate');
    }

    /**
     * Remove the specified resource from storage.
```

```
     */     public function
destroy(string $id)
    {
        $useri= m_user::findOrFail($id)->delete();
return \redirect()->route('m_user.index')

        -> with('success', 'data Berhasil Dihapus');
    }
}
```

7. Buatlah folder di Resources/Views/**m_user** dengan beberapa blade dan isian kode berikut:

1. template.blade.php

```
<!DOCTYPE html>
<html>
<head>
<title>CRUD Laravel</title> <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.m
i n.css"> </head>
<body>
<div class="container">
@yield('content')
</div>
</body>
</html>
```

2.index.blade.php

```
@extends('m_user/template')
@section('content')
<div class="row mt-5 mb-5">
<div class="col-lg-12 margin-tb">
<div class="float-left">
<h2>CRUD user</h2>
</div>
<div class="float-right">
<a class="btn btn-success" href="{{ route('m_user.create') }}"> Input
User</a>
```

```
</div>
</div>
</div>
@if ($message = Session::get('success'))
<div class="alert alert-success">
<p>{{ $message }}</p>
</div>
@endif
<table class="table table-bordered">
<tr>
<th width="20px" class="text-center">User id</th>
<th width="150px" class="text-center">Level id</th>
<th width="200px"class="text-
center">username</th> <th
width="200px"class="text-center">nama</th>
<th width="150px"class="text-center">password</th>
</tr>
@foreach ($useri as $m_user)
<tr>

<td>{{ $m_user->user_id }}</td>
<td>{{ $m_user->level_id }}</td>
<td>{{ $m_user->username }}</td>
<td>{{ $m_user->nama }}</td>
<td>{{ $m_user->password }}</td>


<td class="text-center">
<form action="{{ route('m_user.destroy',$m_user->user_id) }}"
method="POST">
<a class="btn btn-info btn-sm" href="{{
route('m_user.show',$m_user-
>user_id) }}">Show</a>
<a class="btn btn-primary btn-sm" href="{{
route('m_user.edit',$m_user->user_id) }}">Edit</a>
@csrf
@method('DELETE')
<button    type="submit"    class="btn    btn-danger    btn-sm"
onclick="return confirm('Apakah Anda yakin ingin menghapus data
ini?')">Delete</button>
</form>
</td>
</tr>
@endforeach
</table>
```

```
@endsection
```

3. create.blade.php

```
@extends('m_user/template')
@section('content')
<div class="row mt-5 mb-5">
<div class="col-lg-12 margin-tb">
<div class="float-left">
<h2>Membuat Daftar User</h2>
</div>
<div class="float-right">
<a class="btn btn-secondary" href="{{
route('m_user.index') }}">
Kembali</a>
</div>
</div>
</div>
@if ($errors->any())
<div class="alert alert-danger">
<strong>Ops</strong> Input gagal<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<form action="{{ route('m_user.store') }}"
method="POST">
@csrf



<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>Username:</strong>
<input type="text" name="username" class="form-control"
placeholder="Masukkan username"></input>
</div>
</div>

<div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
    <strong>nama:</strong>
    <input type="text" name="nama" class="form-
control" placeholder="Masukkan nama"></input>
    </div>
```

```
    </div>


<div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
    <strong>Password:</strong>
    <input type="password" name="password" class="form-
control" placeholder="Masukkan password"></input>
    </div>
    </div>
<div class="col-xs-12 col-sm-12 col-md-12 text-
center">
<button type="submit" class="btn btn-
primary">Submit</button>
</div>
</div>
</form>
@endsection
```

4.show.blade.php

```
@extends('m_user/template')
@section('content')
<div class="row mt-5 mb-5">
<div class="col-lg-12 margin-tb">
<div class="float-left">
<h2> Show User</h2>
</div>
<div class="float-right">
<a class="btn btn-secondary" href="{{ route('m_user.index') }}">
Kembali</a>
</div>
</div>
</div>
<div class="row">
<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>User_id:</strong>
{{ $useri->user_id }}
</div>
</div>
<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>Level_id:</strong>
{{ $useri->level_id }}
</div>
```

```
</div>

<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>Username:</strong>
{{ $useri->username }}
</div>
</div>


<div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
    <strong>Nama:</strong>
    {{ $useri->nama }}
    </div>
    </div>

<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>Password:</strong>
{{ $useri->password }}
</div>
</div>
</div>
@endsection
```

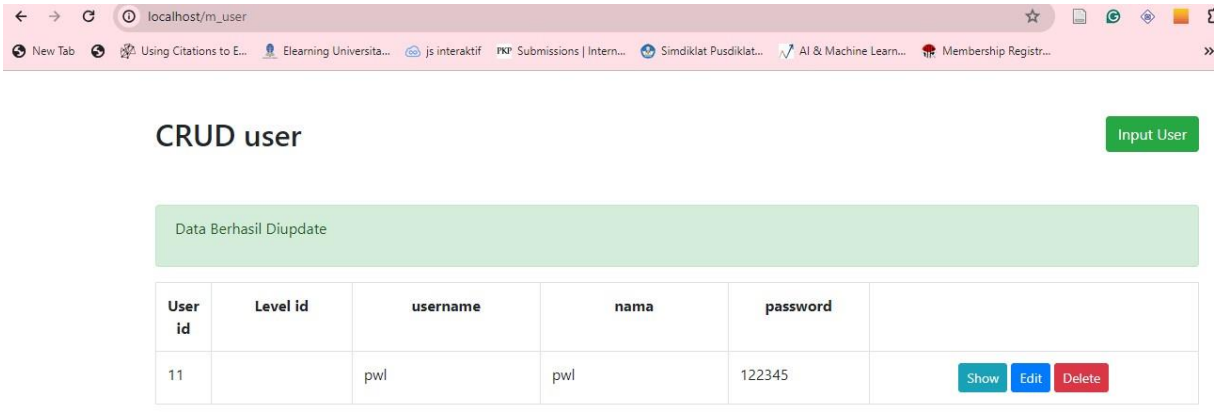
5. edit.blade.php

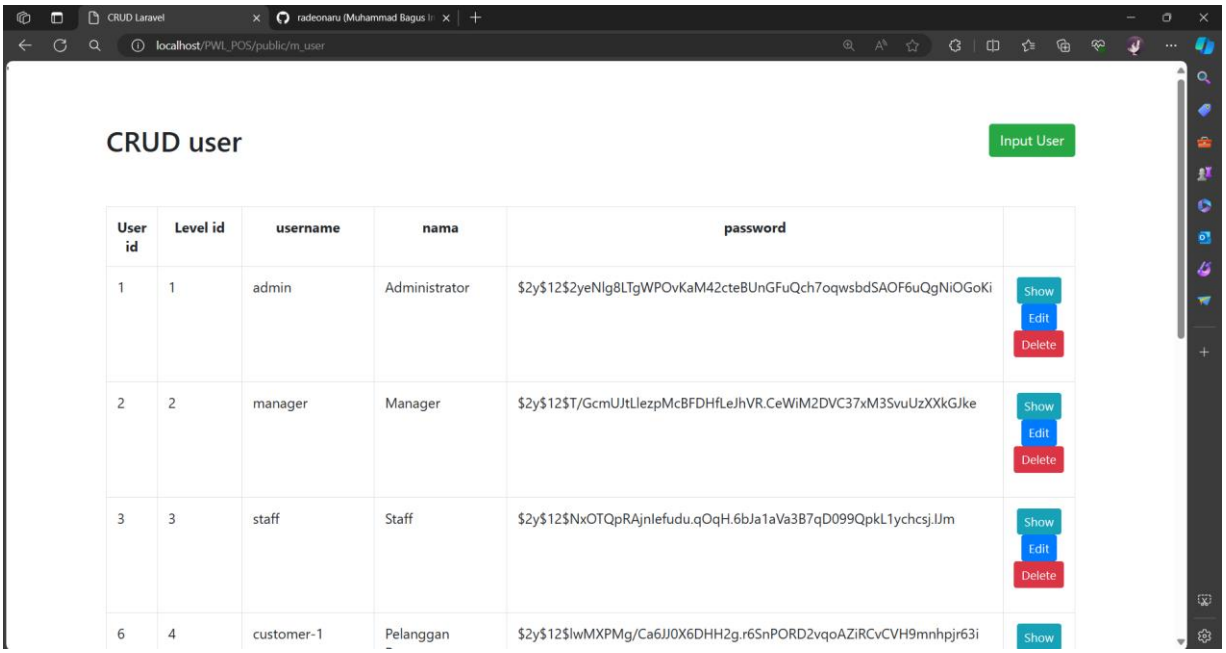
```
@extends('m_user/template')
@section('content')
<div class="row mt-5 mb-5">
<div class="col-lg-12 margin-tb">
<div class="float-left">
<h2>Edit User</h2>
</div>
<div class="float-right">
<a class="btn btn-secondary" href="{{ route('m_user.index') }}">
Kembali</a>
</div>
</div>
</div>
@if ($errors->any())
<div class="alert alert-danger">
```

```
<strong>Ops!</strong> Error <br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<form action="{{ route('m_user.update', $useri->user_id) }}"
method="POST"> @csrf
@method('PUT')
<div class="row">
    <div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
    <strong>User_id:</strong>
    <input type="text" name="userid" value="{{ $useri->user_id }}"
class="form-control" placeholder="Masukkan user id">
    </div>
    </div>

<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>Level_id:</strong>
<input type="text" name="levelid" value="{{ $useri->level_id }}"
class="form-control" placeholder="Masukkan level">
</div>
</div>

<div class="col-xs-12 col-sm-12 col-md-12">
<div class="form-group">
<strong>Username:</strong>
<input type="text" value= "{{ $useri->username }}"
class="formcontrol" name="username" placeholder="Masukkan Nomor
username"">
</div>
</div>

<div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
    <strong>nama:</strong>
    <input type="text" value= "{{ $useri->nama }}"name="nama"
class="form-control" placeholder="Masukkan nama"></input>
    </div>
    </div>
```

```
<div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
    <strong>Password:</strong>
    <input type="password" value= "{{ $useri->password
}}"name="password" class="form-control" placeholder="Masukkan
password"></input>
    </div>
    </div>


<div class="col-xs-12 col-sm-12 col-md-12 text-center">
<button type="submit" class="btn btn-primary">Update</button>
</div>
</div>
</form>
@endsection
```

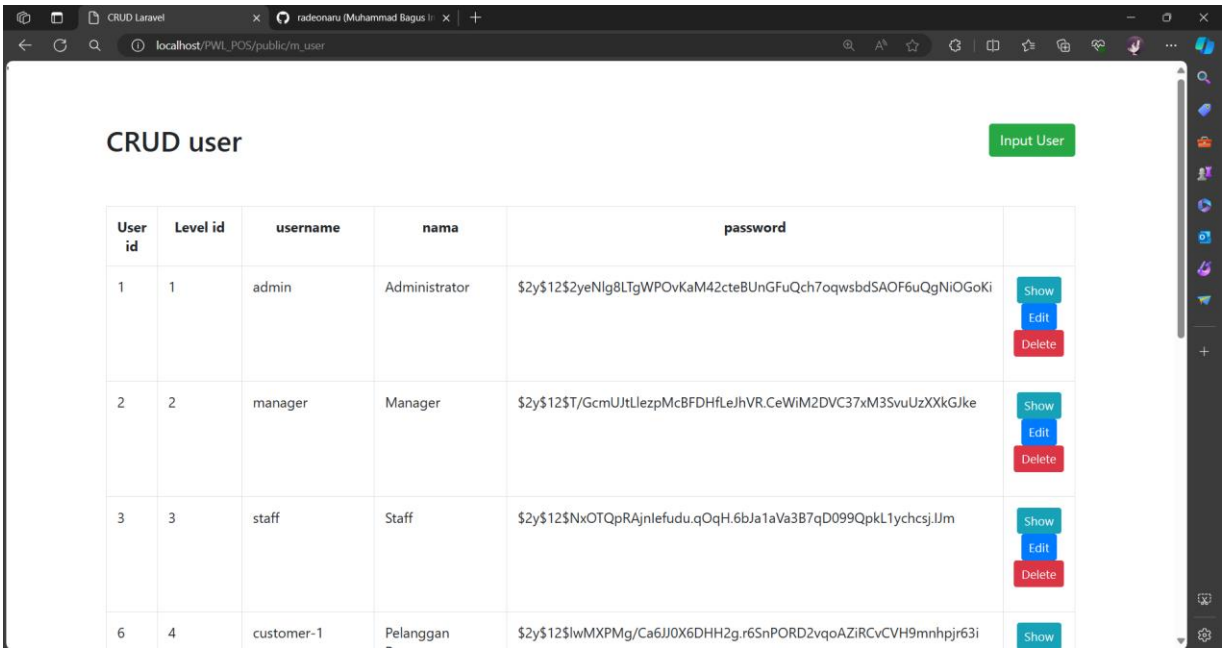
Silahkan akses localhost/m_user



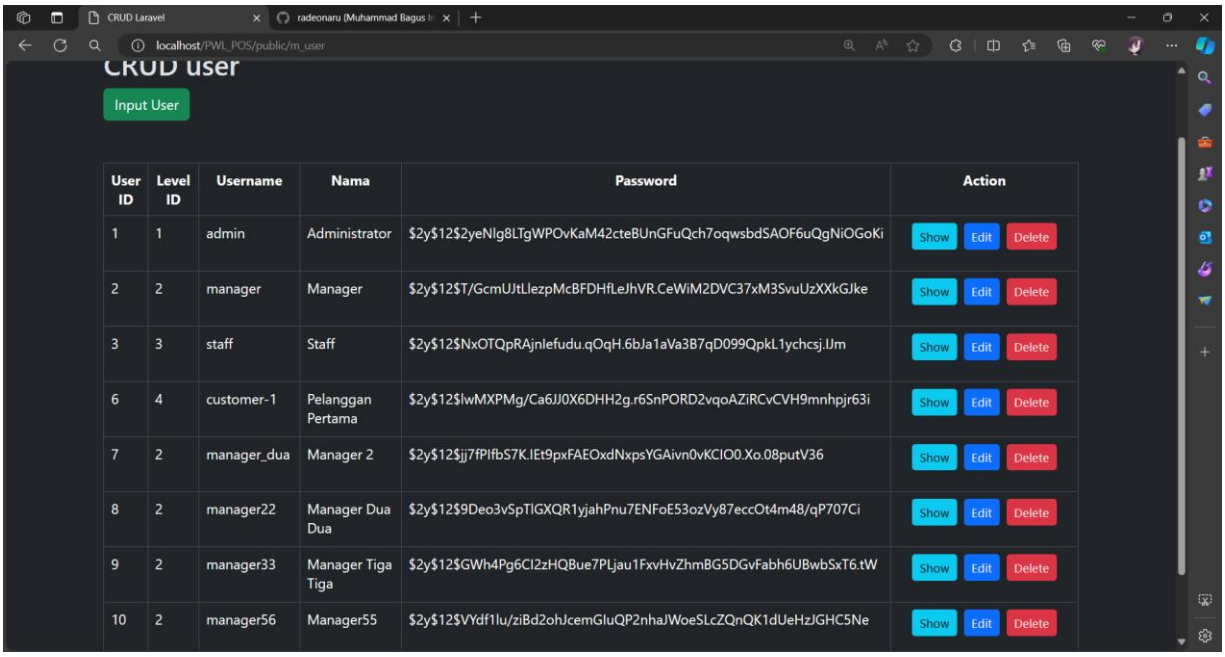


Tugas:

1. Coba tampilkan level_id pada halaman web tersebut dimana field ini merupakan foreign key



2. Modifikasi dengan tema/ template kesukaan Anda



3. Apa fungsi `$request->validate`, $error dan alert yang ada pada halaman CRUD tersebut

- $request->validate: $request->validate adalah metode yang digunakan untuk melakukan validasi pada data yang dikirim melalui HTTP request. Ketika kita menerima data dari form (misalnya saat menyimpan data melalui form input), kita dapat menggunakan $request->validate untuk memastikan bahwa data tersebut memenuhi kriteria validasi yang telah ditentukan. Jika validasi gagal, Laravel akan secara otomatis mengarahkan pengguna kembali ke halaman form dengan pesan kesalahan yang relevan.

- $errors: $errors digunakan untuk memberikan informasi tentang kesalahan validasi yang terjadi. Setelah validasi gagal, kita dapat mengakses $errors untuk menampilkan pesan kesalahan secara spesifik.

- alert: alert digunakan untuk menampilkan pesan kepada pengguna. Ketika kita ingin memberikan informasi atau peringatan kepada pengguna (misalnya setelah berhasil atau gagal menyimpan data), kita dapat menggunakan alert.

**** Sekian, dan selamat belajar ****