



Mata Kuliah : Pemrograman Web Lanjut (PWL)  
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis  
Semester : 4 (empat) / 5 (lima)  
Pertemuan ke- : 6 (enam)

## JOBSHEET 06

### Ajax Form (AdminLTE) dan Client Validation

Proses pembuatan form CRUD (Create, Read, Update, Delete) dengan validasi di Laravel 10 menggunakan jQuery Validation melibatkan beberapa langkah penting yang mencakup pengaturan database, pembuatan model dan migrasi, pengembangan controller, penulisan view, dan penambahan validasi form di sisi klien. *Client side form 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.

Salah satu cara yang populer untuk melakukan validasi di sisi klien adalah dengan menggunakan plugin jQuery Validation. Plugin **jQuery Validation** digunakan untuk menambahkan validasi sisi klien pada form. Misalnya, Kita bisa mengatur agar suatu input wajib diisi dan tidak boleh lebih dari 255 karakter. Validasi ini membantu dalam memberikan umpan balik langsung kepada pengguna tentang kesalahan input tanpa perlu memuat ulang halaman ataupun mengirim *request* ke server.

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

#### A. AJAX form

**AJAX** (Asynchronous JavaScript and XML) adalah sebuah teknik atau metode dalam pengembangan web yang memungkinkan aplikasi web untuk mengirim dan menerima data dari server secara asinkron (tanpa memuat ulang seluruh halaman). Dengan AJAX, interaksi antara



klien dan server menjadi lebih dinamis dan responsif, karena pengguna dapat berinteraksi dengan halaman web dan melihat perubahan langsung tanpa harus melakukan refresh halaman.

*Ajax form* adalah teknik di mana sebuah form HTML dikirim ke server secara asinkron menggunakan AJAX, tanpa memuat ulang seluruh halaman web. Dengan AJAX form, Kita bisa mengirim data ke server dan menampilkan respons secara dinamis di halaman, sehingga meningkatkan pengalaman pengguna dengan membuat interaksi lebih cepat dan lebih responsif.

### **Mengapa Menggunakan AJAX Form?**

1. **Response Instan:** AJAX memungkinkan Kita untuk mengirim data dan menerima respons dari server tanpa perlu memuat ulang halaman.
2. **Pengalaman Pengguna yang Lebih Baik:** Karena tidak ada pemuatan ulang halaman, aplikasi terasa lebih cepat dan lebih interaktif, mirip dengan aplikasi desktop.

**Pengurangan Beban Server:** Dengan mengirim hanya data yang diperlukan, AJAX dapat mengurangi penggunaan bandwidth dan beban di server.

## **B. Validasi Sisi Client**

**Validasi di sisi klien** adalah proses pemeriksaan data yang dimasukkan oleh pengguna pada form web sebelum data tersebut dikirim ke server. Validasi ini dilakukan menggunakan kode yang berjalan di browser pengguna, seperti JavaScript, dan bertujuan untuk memastikan bahwa data yang dimasukkan sesuai dengan aturan tertentu, seperti format email yang benar, panjang karakter yang sesuai, atau tidak adanya kolom kosong yang wajib diisi.

### **Tujuan dan Manfaat Validasi di Sisi Klien**

#### **1. Umpan Balik Instan**

Pengguna mendapatkan umpan balik segera setelah mereka memasukkan data yang tidak valid, seperti kesalahan format email atau kolom yang tidak diisi. Ini meningkatkan pengalaman pengguna (*user experience*) karena mereka tidak perlu menunggu respon dari server untuk mengetahui apakah input mereka benar atau salah.

#### **2. Mengurangi Beban Server**

Dengan melakukan validasi di sisi klien, kesalahan dapat diidentifikasi dan diperbaiki sebelum data dikirim ke server, sehingga server tidak perlu memproses permintaan yang tidak valid. Ini dapat mengurangi beban kerja server dan meningkatkan kinerja aplikasi.

#### **3. Meningkatkan Efisiensi**



Validasi di sisi klien membantu mencegah pengiriman data yang tidak valid, sehingga mengurangi jumlah permintaan HTTP yang perlu diproses oleh server. Hal ini menghemat bandwidth dan waktu pemrosesan, membuat aplikasi lebih efisien.

#### 4. **Memastikan Integritas Data**

Dengan validasi sisi klien, banyak kesalahan input yang dapat dicegah sebelum data mencapai server. Misalnya, memastikan bahwa nomor telepon hanya berisi angka atau alamat email mengikuti format yang benar.

#### 5. **Menyederhanakan Proses Pengembangan**

Dengan validasi di sisi klien, pengembang dapat menangani banyak potensi kesalahan input di awal, yang menyederhanakan logika pemrosesan di sisi server. Ini memungkinkan pengembang untuk fokus pada validasi yang lebih kompleks atau logika bisnis lainnya di server.

#### 6. **Meningkatkan Keamanan**

Meskipun validasi sisi klien tidak bisa menggantikan validasi di sisi server (karena dapat dengan mudah diabaikan atau dimanipulasi oleh pengguna yang berpengalaman), validasi ini tetap dapat membantu dalam mengurangi jumlah data yang tidak valid yang mencapai server. Ini berfungsi sebagai lapisan pertama pertahanan, mencegah beberapa jenis input yang tidak diinginkan.

#### 7. **Memberikan Panduan Pengguna**

Validasi sisi klien memungkinkan pengembang untuk memberikan panduan dan instruksi yang lebih baik kepada pengguna tentang cara memasukkan data dengan benar. Misalnya, pesan kesalahan bisa ditampilkan di bawah kolom yang salah, memberikan petunjuk spesifik kepada pengguna

### **Bagaimana Validasi di Sisi Klien Bekerja?**

Validasi di sisi klien biasanya dilakukan menggunakan JavaScript atau framework JavaScript seperti jQuery. Berikut adalah contoh sederhana validasi form di sisi klien:

```
<!DOCTYPE html>
<html>
<head>
  <title>Client-Side Validation Example</title>
</head>
<body>
  <form name="myForm" onsubmit="return validateForm()" method="post">
    Name: <input type="text" name="name"><br><br>
    Email: <input type="text" name="email"><br><br>
    <input type="submit" value="Submit">
  </form>
  <script>
    function validateForm() {
      var email = document.forms["myForm"]["email"].value;
      var name = document.forms["myForm"]["name"].value;
```



```
if (name == "") {  
    alert("Name must be filled out");  
    return false;  
}  
  
var emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;  
if (!emailPattern.test(email)) {  
    alert("Please enter a valid email address");  
    return false;  
}  
return true;  
}  
</script>  
</body>  
</html>
```

### Keuntungan Validasi di Sisi Klien

#### 1. Responsif

Pengguna mendapatkan respons cepat terhadap input mereka tanpa perlu menunggu interaksi dengan server.

#### 2. Interaktif

Dapat memberikan instruksi tambahan dan lebih kontekstual kepada pengguna untuk memperbaiki kesalahan.

#### 3. Penghematan Sumber Daya

Mengurangi jumlah permintaan ke server yang tidak perlu, sehingga menghemat bandwidth dan sumber daya server.

### Keterbatasan Validasi di Sisi Klien

#### 1. Tidak Mengganti Validasi di Sisi Server

Validasi di sisi klien dapat dilewati oleh pengguna berpengalaman atau perangkat otomatis. Oleh karena itu, validasi di sisi klien harus selalu dilengkapi dengan validasi di sisi server untuk memastikan keamanan dan integritas data.

#### 2. Ketergantungan pada JavaScript

Jika pengguna menonaktifkan JavaScript di browser mereka, validasi di sisi klien tidak akan berfungsi.

Validasi di sisi klien merupakan komponen penting dalam pengembangan aplikasi web modern, karena meningkatkan pengalaman pengguna dan efisiensi aplikasi. Namun, ini harus selalu digunakan bersama dengan validasi di sisi server untuk menjaga keamanan dan memastikan data yang diterima oleh aplikasi adalah valid dan sesuai dengan aturan bisnis.



### C. jQuery Validation

Salah satu cara yang populer untuk melakukan validasi di sisi klien adalah dengan menggunakan plugin jQuery Validation. **jQuery Validation** adalah plugin jQuery yang digunakan untuk memvalidasi form HTML di sisi klien secara efisien dan interaktif. Plugin ini memudahkan pengembang untuk menambahkan logika validasi pada form dengan cara yang mudah dan dapat disesuaikan, memberikan umpan balik langsung kepada pengguna mengenai kesalahan input mereka sebelum data dikirim ke server.

#### Fitur Utama jQuery Validation

##### 1. Kemudahan Penggunaan

jQuery Validation dirancang untuk memudahkan integrasi dan penggunaan. Dengan beberapa baris kode, Kita dapat menambahkan validasi ke form HTML tanpa perlu menulis logika validasi dari awal.

##### 2. Validasi Real-Time

Plugin ini memvalidasi input form secara real-time saat pengguna mengetik atau setelah mereka pindah dari satu field ke field lainnya. Ini memberikan umpan balik langsung kepada pengguna mengenai kesalahan input mereka.

##### 3. Aturan Validasi yang Siap Pakai:

jQuery Validation menyediakan berbagai aturan validasi yang siap digunakan, seperti:

- *required*: Memastikan bahwa field tidak kosong.
- *email*: Memastikan bahwa input berformat alamat email yang valid.
- *url*: Memastikan bahwa input berformat URL yang valid.
- *minlength* dan *maxlength*: Membatasi jumlah karakter minimum atau maksimum dalam input.
- *number*: Memastikan bahwa input hanya berisi angka.

##### 4. Pesan Kesalahan Kustom

Kita dapat menyesuaikan pesan kesalahan yang ditampilkan kepada pengguna. Misalnya, Kita dapat mengubah pesan default seperti "This field is required" menjadi sesuatu yang lebih spesifik atau sesuai dengan konteks aplikasi Kita.

##### 5. Integrasi dengan jQuery UI

jQuery Validation dapat dengan mudah diintegrasikan dengan jQuery UI untuk menampilkan pesan kesalahan dalam format yang lebih menarik, seperti menggunakan tooltip atau dialog box.

##### 6. Validasi Multi-Field



Plugin ini mendukung validasi yang melibatkan lebih dari satu field. Misalnya, Kita bisa memastikan bahwa dua field password dan konfirmasi password memiliki nilai yang sama.

## 7. Plugin dan Ekstensi

jQuery Validation memiliki ekosistem plugin dan ekstensi yang memungkinkan Kita menambahkan aturan validasi kustom atau mengubah perilaku default.

### Cara Menggunakan jQuery Validation

Berikut adalah contoh sederhana bagaimana jQuery Validation digunakan untuk memvalidasi form:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>jQuery Validation Example</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.1/jquery-
  ui.css">
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-
  validate/1.19.3/jquery.validate.min.js"></script>
</head>
<body>
  <form id="myForm">
    <label for="name">Name:</label><input type="text" name="name" id="name"><br>
    <label for="email">Email:</label><input type="text" name="email" id="email"><br>
    <input type="submit" value="Submit">
  </form>

  <script>
    $(document).ready(function() {
      $("#myForm").validate({
        rules: {
          name: "required",
          email: {
            required: true,
            email: true
          }
        },
        messages: {
          name: "Please enter your name",
          email: "Please enter a valid email address"
        }
      });
    });
  </script>
</body>
</html>
```

### Penjelasan:

- **rules**: mendefinisikan aturan validasi untuk setiap field. Dalam contoh di atas:
  - *Field name* harus diisi (required).
  - *Field email* harus diisi dan harus berformat email yang valid (email).



- **messages:** mendefinisikan pesan kesalahan yang akan ditampilkan jika aturan validasi tidak terpenuhi.

### Keuntungan Menggunakan jQuery Validation

1. **Pengalaman Pengguna yang Lebih Baik**  
Pengguna mendapatkan umpan balik langsung, yang membantu mereka memperbaiki kesalahan input dengan cepat.
2. **Pengurangan Beban Server**  
Validasi di sisi klien mengurangi jumlah permintaan yang tidak valid yang dikirim ke server, menghemat sumber daya server.
3. **Fleksibilitas dan Kustomisasi**  
Plugin ini sangat fleksibel dan dapat dikustomisasi sesuai kebutuhan aplikasi, dari aturan validasi hingga pesan kesalahan yang ditampilkan.
4. **Kompatibilitas dengan Semua Browser Modern**  
jQuery Validation kompatibel dengan hampir semua browser modern, sehingga dapat digunakan di berbagai lingkungan pengguna.

jQuery Validation memiliki berbagai metode bawaan yang sangat berguna untuk memvalidasi form di sisi klien. Selain metode standar seperti `required`, `email`, dan `number`, Kita juga dapat menambahkan metode validasi kustom menggunakan `addMethod`. Ini memungkinkan Kita untuk membuat aturan validasi yang lebih spesifik sesuai kebutuhan aplikasi Kita.

### D. Method jQuery Validation

jQuery Validation menyediakan beberapa metode bawaan (built-in methods) yang dapat digunakan untuk memvalidasi form dengan berbagai jenis aturan. Selain itu, jQuery Validation juga memungkinkan pengembang untuk menambahkan metode kustom dengan `addMethod`, seperti yang telah dijelaskan sebelumnya. Berikut adalah beberapa metode tambahan yang tersedia dalam jQuery Validation

No	Method	Deskripsi
1	<code>required</code>	<ul style="list-style-type: none"><li>• Memastikan bahwa field tidak kosong.</li><li>• Contoh: <code>required: true</code></li></ul>
2	<code>email</code>	<ul style="list-style-type: none"><li>• Memastikan bahwa input berformat alamat email yang valid.</li><li>• Contoh: <code>email: true</code></li></ul>
3	<code>Url</code>	<ul style="list-style-type: none"><li>• Memastikan bahwa input berformat URL yang valid.</li><li>• Contoh: <code>url: true</code></li></ul>





4	<i>date</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input berformat tanggal yang valid (berdasarkan pengaturan regional)</li><li>Contoh: <code>date: true</code></li></ul>
5	<i>dateISO</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input berformat tanggal yang valid dalam format ISO (YYYY-MM-DD)</li><li>Contoh: <code>dateISO: true</code></li></ul>
6	<i>number</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input hanya berisi angka (integer atau desimal).</li><li>Contoh: <code>number: true</code></li></ul>
7	<i>digits</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input hanya berisi angka (tanpa desimal).</li><li>Contoh: <code>digits: true</code></li></ul>
8	<i>creditcard</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input berformat nomor kartu kredit yang valid.</li><li>Contoh: <code>creditcard: true</code></li></ul>
9	<i>equalTo</i>	<ul style="list-style-type: none"><li>Memastikan bahwa nilai elemen form sama dengan elemen lain (misalnya, untuk konfirmasi password).</li><li>Contoh: <code>equalTo: "#password"</code></li></ul>
10	<i>maxlength</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input tidak melebihi jumlah karakter tertentu.</li><li>Contoh: <code>maxlength: 10</code></li></ul>
11	<i>minlength</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input memiliki minimal jumlah karakter tertentu.</li><li>Contoh: <code>minlength: 5</code></li></ul>
12	<i>rangelength</i>	<ul style="list-style-type: none"><li>Memastikan bahwa panjang input berada dalam rentang karakter tertentu.</li><li>Contoh: <code>rangelength: [5, 10]</code></li></ul>
13	<i>range</i>	<ul style="list-style-type: none"><li>Memastikan bahwa nilai input berada dalam rentang tertentu (misalnya, angka 1 sampai 100)</li><li>Contoh: <code>range: [1, 100]</code></li></ul>
14	<i>max</i>	<ul style="list-style-type: none"><li>Memastikan bahwa nilai input tidak melebihi angka maksimum tertentu.</li><li>Contoh: <code>max: 100</code></li></ul>
15	<i>min</i>	<ul style="list-style-type: none"><li>Memastikan bahwa nilai input tidak kurang dari angka minimum tertentu.</li><li>Contoh: <code>min: 1</code></li></ul>
16	<i>remote</i>	<ul style="list-style-type: none"><li>Memvalidasi nilai dengan mengirimkan permintaan ke server untuk memeriksa apakah nilai tersebut valid atau tersedia (misalnya, memeriksa ketersediaan username)</li><li>Contoh<pre>remote: {   url: "/check-username",   type: "post" }</pre></li></ul>
17	<i>step</i>	<ul style="list-style-type: none"><li>Memastikan bahwa nilai input adalah kelipatan dari angka tertentu (berguna untuk validasi angka desimal).</li><li>Contoh: <code>step: 10</code></li></ul>
18	<i>phoneUS</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input berformat nomor telepon yang valid di AS.</li><li>Contoh: <code>phoneUS: true</code></li></ul>





19	<i>extension</i>	<ul style="list-style-type: none"><li>Memastikan bahwa file yang diupload memiliki ekstensi tertentu.</li><li>Contoh: <code>extension: "jpg png gif"</code></li></ul>
20	<i>accept</i>	<ul style="list-style-type: none"><li>Memastikan bahwa file yang diupload memiliki jenis MIME tertentu.</li><li>Contoh: <code>accept: "image/*"</code></li></ul>
21	<i>exactlength</i>	<ul style="list-style-type: none"><li>Memastikan bahwa input hanya berisi karakter yang panjangnya sama persis dengan ketentuan.</li><li>Contoh: <code>exactlength: 10</code></li></ul>

**jQuery Validation** adalah alat yang sangat berguna untuk memastikan data yang dimasukkan ke dalam form web valid dan sesuai dengan aturan yang ditetapkan sebelum data tersebut dikirim ke server. Ini meningkatkan pengalaman pengguna, mengurangi kesalahan, dan mempermudah pengelolaan validasi form di sisi klien dalam pengembangan aplikasi web.

## E. Praktikum Jobsheet

Langsung saja kita praktikkan untuk menggunakan Ajax form dan validasi disisi client.

### Praktikum 1. Modal Ajax Tambah Data (Data User)

- Kita buat form tambah data baru dengan menerapkan modal dan proses ajax.
- Pertama yang kita siapkan adalah menambahkan *library jQuery Validation* dan *Sweetalert* ke aplikasi web kita. Caranya kita tambahkan link kedua *library* tersebut ke `template.blade.php`, library sudah disediakan oleh adminLTE.

```
template.blade.php · X
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>AdminLTE 3 | Blank Page</title>
7
8      <meta name="csrf-token" content="{{ csrf_token() }}">
9
10     <!-- Font Awesome -->
11     <link rel="stylesheet" href="{{ asset('plugins/fontawesome-free/css/all.min.css') }}">
12     <!-- DataTables -->
13     <link rel="stylesheet" href="{{ asset('plugins/datatables-bs4/css/dataTables.bootstrap4.min.css') }}">
14     <link rel="stylesheet" href="{{ asset('plugins/datatables-responsive/css/responsive.bootstrap4.min.css') }}">
15     <link rel="stylesheet" href="{{ asset('plugins/datatables-buttons/css/buttons.bootstrap4.min.css') }}">
16     <!-- SweetAlert2 -->
17     <link rel="stylesheet" href="{{ asset('plugins/sweetalert2-theme-bootstrap-4/bootstrap-4.min.css') }}">
18     <!-- Theme style -->
19     <link rel="stylesheet" href="{{ asset('dist/css/adminlte.min.css') }}">
20
21     @stack('css')
22 </head>
```

```
18  {{-- sweet alert 2 --}}
19  <link rel="stylesheet" href="{{ asset('adminlte/plugins/sweetalert2-theme-bootstrap-4.min.css') }}">
20
```



```
template.blade.php X
65 <script src="{{ asset('plugins/pdfmake/pdfmake.min.js') }}"></script>
66 <script src="{{ asset('plugins/pdfmake/vfs_fonts.js') }}"></script>
67 <script src="{{ asset('plugins/datatables-buttons/js/buttons.html5.min.js') }}"></script>
68 <script src="{{ asset('plugins/datatables-buttons/js/buttons.print.min.js') }}"></script>
69 <script src="{{ asset('plugins/datatables-buttons/js/buttons.colVis.min.js') }}"></script>
70
71 <!-- jquery-validation -->
72 <script src="{{ asset('plugins/jquery-validation/jquery.validate.min.js') }}"></script>
73 <script src="{{ asset('plugins/jquery-validation/additional-methods.min.js') }}"></script>
74
75 <!-- SweetAlert2 -->
76 <script src="{{ asset('plugins/sweetalert2/sweetalert2.min.js') }}"></script>
77
78 <!-- AdminLTE App -->
79 <script src="{{ asset('dist/js/adminlte.min.js') }}"></script>
80 <script>
81     $.ajaxSetup({
82         headers: {
83             'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
84         }
85     });
86 </script>
87
88 @stack('js')
89 </body>
90 </html>
80
81 {{-- jQuery validation --}}
82 <script src="{{asset('adminlte/plugins/jquery-validation/jquery.validate.min.js')}}"></script>
83 <script src="{{asset('adminlte/plugins/jquery-validation/additional-methods.min.js')}}"></script>
84
85 {{--/sweetalert2 --}}
86 <script src="{{asset('adminlte/plugins/sweetalert2/sweetalert2.min.js')}}"></script>
87
```

3. Selanjutnya Kita modifikasi view [user/index.blade.php](#), kita tambahkan tombol untuk membuat form popup ajax



```
@extends('layouts.template')

@section('content')
    <div class="card card-outline card-primary">
        <div class="card-header">
            <h3 class="card-title">{{ $page->title }}</h3>
            <div class="card-tools">
                <a class="btn btn-sm btn-primary mt-1" href="{{ url('user/create') }}">Tambah</a>
                <button onclick="modalAction('{{ url('user/create_ajax') }}')" class="btn btn-sm btn-success mt-1">Tambah Ajax</button>
            </div>
        </div>
        <div class="card-body">
            @if (@session('success'))
                <div class="alert alert-success">{{ session('success') }}</div>
            @endif
            @if (@session('error'))
                <div class="alert alert-danger">{{ session('error') }}</div>
            @endif
            <div class="row">
                <div class="col-md-12">
                    <div class="form-group row">
                        <label class="col-1 control-label col-form-label">Filter:</label>
                        <div class="col-3">
                            <select class="form-control" id="level_id" name="level_id" required>
                                <option value="">- Semua -</option>
                                @foreach ($level as $item)
                                    <option value="{{ $item->level_id }}">{{ $item->level_nama }}</option>
                                @endforeach
                            </select>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

Kita tambahkan kode berikut, untuk membuat form modal tambah data user dengan ajax

```
Minggu_6 > PWL_POS > resources > views > user > index.blade.php > div.card.card-outline.card-primary > script > <function> > <function>
1  @extends('layouts.template')
2  @section('content')
3      <div class="card card-outline card-primary">
4          <div class="card-header">
5              <h3 class="card-title">{{ $page->title }}</h3>
6              <div class="card-tools">
7                  <a class="btn btn-sm btn-primary mt-1" href="{{ url('user/create') }}">Tambah</a>
8                  <button onclick="modalAction('{{ url('user/create_ajax') }}')" class="btn btn-sm btn-success mt-1">Tambah
9                      Ajax</button>
10             </div>
11         </div>
12         <div class="card-body">
```



```
<button onclick="modalAction('{{ url('/user/create_ajax') }}')" class="btn btn-sm btn-success mt-1">Tambah Ajax</button>
```

4. Selanjutnya kita tambahkan kode berikut pada **akhir** `@section('content')` pada view `user/index.blade.php`

```
<div id="myModal" class="modal fade animate shake" tabindex="-1" role="dialog" data-backdrop="static" data-keyboard="false" data-width="75%" aria-hidden="true"></div>
```

```
47 <div id="myModal" class="modal fade animate shake" tabindex="-1" role="dialog" data-backdrop="static"
48 data-keyboard="false" data-width="75%" aria-hidden="true"></div>
49 @endsection
```

5. Kemudian kita tambahkan kode berikut pada **awal** `@push('js')` pada view `user/index.blade.php`

```
function modalAction(url = ''){
    $('#myModal').load(url,function(){
        $('#myModal').modal('show');
    });
}
```

```
52 @push('js')
53 <script>
54     function modalAction(url = '') {
55         $('#myModal').load(url, function() {
56             $('#myModal').modal('show');
57         });
58     }
59     var dataUser;
60     $(document).ready(function() {
61         dataUser = $('#table_user').DataTable({
62             // serverSide: true, jika ingin menggunakan server side processing
63             serverSide: true,
64             ajax: {
65                 "url": "{{ url('user/list') }}",
66                 "dataType": "json",
67                 "type": "POST",
68                 "data": function(d) {
69                     d.level_id = $('#level_id').val();
70                 }
71             },
```



Sehingga tampilan kode program akan seperti berikut

```
<div id="myModal" class="modal fade animate shake" tabindex="-1" role="dialog" data-backdrop="static"
data-keyboard="false" data-width="75%" aria-hidden="true"></div>
@endsection

@push('css')
@endpush

@push('js')
<script>
    function modalAction(url = ''){
        $('#myModal').load(url,function(){
            $('#myModal').modal('show');
        });
    }

    var dataUser;
    $(document).ready(function() {
        dataUser = $('#table_user').DataTable({
            // serverSide: true, jika ingin menggunakan server side processing
            serverSide: true,
            ajax: {
                "url": "{{ url('user/list') }}",
                "dataType": "json",
                "type": "POST",
                "data": function (d){
                    d.level_id = $('#level_id').val();
                }
            },
            columns: [ {
```

Ubah seperti ini

6. Selanjutnya Kita modifikasi [route/web.php](#) untuk mengakomodir operasi ajax





```
Route::group(['prefix' => 'user'], function () {
    Route::get('/', [UserController::class, 'index']); // Menampilkan halaman awal user
    Route::post('/list', [UserController::class, 'list']); // Menampilkan data user dalam bentuk json untuk datatables
    Route::get('/create', [UserController::class, 'create']); // Menampilkan halaman form tambah user
    Route::post('/', [UserController::class, 'store']); // Menyimpan data user baru
    Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
    Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax
    Route::get('/{id}', [UserController::class, 'show']); // Menampilkan detail user
    Route::get('/{id}/edit', [UserController::class, 'edit']); // Menampilkan halaman form edit user
    Route::put('/{id}', [UserController::class, 'update']); // Menyimpan perubahan data user
    Route::delete('/{id}', [UserController::class, 'destroy']); // Menghapus data user
});
```

```
47 Route::get('/create_ajax', [UserController::class, 'create_ajax']); // menampilkan bentuk form untuk tambah user
48 Route::post('/ajax', [UserController::class, 'store_ajax']); // menyimpan user data baru ajax
```

7. Kemudian Kita tambahkan fungsi `create_ajax()` pada file `UserController.php`

```
public function create_ajax()
{
    $level = LevelModel::select('level_id', 'level_nama')->get();

    return view('user.create_ajax')
        ->with('level', $level);
}
```

```
Mykings > PWL_P06 > app > Http > Controllers > UserController.php > UserController > store_ajax
11 use Mockery\Mocker;
12 use Yajra\DataTables\Facades\DataTables;
13 use Illuminate\Support\Facades\Validator;
14
15 use function Laravel\Prompts\error;
16
17 class UserController extends Controller
18 {
19
20     public function create_ajax()
21     {
22         $level = LevelModel::select('level_id', 'level_nama')->get();
23
24         return View('user.create_ajax')
25             ->with('level', $level);
26     }
27
28     public function store_ajax(Request $request)
29     {
30         // validasi ajax
31         if ($request->isJson() || $request->wantsJson()) {
32             $rules = [
33                 'level_id' => 'required|integer',
34                 'username' => 'required|string|min:3|unique:users,username',
35                 'name' => 'required|string|max:100',
36                 'password' => 'required|min:6',
37             ];
38         }
39     }
40 }
```



8. Setelah itu, kita buat **view baru** `user/create_ajax.blade.php` untuk menampilkan form dengan ajax

```
<form action="{{ url('/user/ajax') }}" method="POST" id="form-tambah">
@csrf
<div id="modal-master" class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
        <div class="modal-header">
            <h5 class="modal-title" id="exampleModallabel">Tambah Data User</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
        </div>
        <div class="modal-body">
            <div class="form-group">
                <label>Level Pengguna</label>
                <select name="level_id" id="level_id" class="form-control" required>
                    <option value="">- Pilih Level -</option>
                    @foreach($level as $l)
                        <option value="{{ $l->level_id }}">{{ $l->level_nama }}</option>
                    @endforeach
                </select>
                <small id="error-level_id" class="error-text form-text text-danger"></small>
            </div>
            <div class="form-group">
                <label>Username</label>
                <input value="" type="text" name="username" id="username" class="form-control"
required>
                <small id="error-username" class="error-text form-text text-danger"></small>
            </div>
            <div class="form-group">
                <label>Nama</label>
                <input value="" type="text" name="nama" id="nama" class="form-control"
required>
                <small id="error-nama" class="error-text form-text text-danger"></small>
            </div>
            <div class="form-group">
                <label>Password</label>
                <input value="" type="password" name="password" id="password" class="form-
control" required>
                <small id="error-password" class="error-text form-text text-danger"></small>
            </div>
        </div>
    </div>
</div>
```





```
<div class="modal-footer">
  <button type="button" data-dismiss="modal" class="btn btn-warning">Batal</button>
  <button type="submit" class="btn btn-primary">Simpan</button>
</div>
</div>
</form>
<script>
$(document).ready(function() {
  $("#form-tambah").validate({
    rules: {
      level_id: {required: true, number: true},
      username: {required: true, minlength: 3, maxlength: 20},
      nama: {required: true, minlength: 3, maxlength: 100},
      password: {required: true, minlength: 6, maxlength: 20}
    },
    submitHandler: function(form) {
      $.ajax({
        url: form.action,
        type: form.method,
        data: $(form).serialize(),
        success: function(response) {
          if(response.status){
            $('#myModal').modal('hide');
            Swal.fire({
              icon: 'success',
              title: 'Berhasil',
              text: response.message
            });
            dataUser.ajax.reload();
          }else{
            $('.error-text').text('');
            $.each(response.msgField, function(prefix, val) {
              $('#error-'+prefix).text(val[0]);
            });
            Swal.fire({
              icon: 'error',
              title: 'Terjadi Kesalahan',
              text: response.message
            });
          }
        }
      });
      return false;
    },
    errorElement: 'span',
    errorPlacement: function (error, element) {
      error.addClass('invalid-feedback');
      element.closest('.form-group').append(error);
    },
    highlight: function (element, errorClass, validClass) {
      $(element).addClass('is-invalid');
    },
    unhighlight: function (element, errorClass, validClass) {
      $(element).removeClass('is-invalid');
    }
  });
});
</script>
```



### Create\_ajax

```
1 <form action="{url}/user/ajax" method="POST" id="form-tambah">
2 </form>
3
4 <div id="modal-master" class="modal-dialog modal-lg" role="document">
5   <div class="modal-content">
6     <div class="modal-header">
7       <div class="modal-title" id="exampleModalLabel">Tambah Data User</div>
8       <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
9         aria-hidden="true"></span></button>
10     </div>
11     <div class="modal-body">
12       <div class="form-group">
13         <label level Pengguna</label>
14         <select name="level_id" id="level_id" class="form-control" required>
15           <option value="">Pilih level</option>
16           <foreach ($level as $i)>
17             <option value="{{ $i->level_id }}" {{ $i->level_name }}</option>
18           </foreach>
19         </select>
20         <small id="error-level_id" class="error-text form-text text-danger"></small>
21       </div>
22       <div class="form-group">
23         <label Username</label>
24         <input value="" type="text" name="username" id="username" class="form-control"
25           required>
26         <small id="error-username" class="error-text form-text text-danger"></small>
27       </div>
28       <div class="form-group">
29         <label Nama</label>
30         <input value="" type="text" name="name" id="name" class="form-control"
31           required>
32         <small id="error-name" class="error-text form-text text-danger"></small>
33       </div>
34       <div class="form-group">
35         <label Password</label>
36         <input value="" type="password" name="password" id="password" class="form-control"
37           required>
38         <small id="error-password" class="error-text form-text text-danger"></small>
39       </div>
40     </div>
41   </div>
42 </div>
```

9. Kemudian untuk mengakomodir proses simpan data melalui ajax, kita buat fungsi `store_ajax()` pada `UserController.php`



```
public function store_ajax(Request $request) {  
    // cek apakah request berupa ajax  
    if($request->ajax() || $request->wantsJson()){  
        $rules = [  
            'level_id' => 'required|integer',  
            'username' => 'required|string|min:3|unique:m_user,username',  
            'nama' => 'required|string|max:100',  
            'password' => 'required|min:6'  
        ];  
  
        // use Illuminate\Support\Facades\Validator;  
        $validator = Validator::make($request->all(), $rules);  
  
        if($validator->fails()){  
            return response()->json([  
                'status' => false, // response status, false: error/gagal, true: berhasil  
                'message' => 'Validasi Gagal',  
                'msgField' => $validator->errors(), // pesan error validasi  
            ]);  
        }  
  
        UserModel::create($request->all());  
        return response()->json([  
            'status' => true,  
            'message' => 'Data user berhasil disimpan'  
        ]);  
    }  
    redirect('/');  
}
```

```
27 public function store_ajax(Request $request){  
28     //cek req ajax  
29     if ($request->ajax()|| $request->wantsJson()) {  
30         $rules =[  
31             'level_id' => 'required|integer',  
32             'username' => 'required|string|min:3|unique:m_user,username',  
33             'nama' => 'required|string|max:100',  
34             'password' => 'required|min:6',  
35         ];  
36     };  
37  
38     $validator = Validator::make($request->all(), $rules);  
39  
40     if ($validator->fails()) {  
41         return response()->json([  
42             'status' => false, //response status  
43             'message' => 'Validasi Gagal', //pesan gagal  
44             'msgField' => $validator->errors(), //pesan error  
45         ]);  
46     }  
47     UserModel::create($request->all());  
48     return response()->json([  
49         'status' => true,  
50         'message' => 'data user berhasil disimpan'  
51     ]);  
52     redirect('/');  
53 }  
54
```



10. OK, sekarang kita coba melakukan proses tambah data user menggunakan form ajax. Amati apa yang terjadi dan laporkan pada laporan *jobsheet* dan *commit* di github kalian!!!

ID	Username	Nama	Level Pengguna	Aksi
1	admin	Administrator	Administrator	Detail Edit Hapus
2	manager	Manager	Manager	Detail Edit Hapus
3	staff	Staff/Kasir	Staff/Kasir	Detail Edit Hapus
4	customer.1	Pelanggan Pertama	Pelanggan	Detail Edit Hapus
5	manager_dua	Manager 2	Manager	Detail Edit Hapus
6	manager22	Manager Dua Dua	Manager	Detail Edit Hapus
7	admin11	admin11	Administrator	Detail Edit Hapus
8	Manager33	Manager33	Manager	Detail Edit Hapus
9	akukk	akukk	Pelanggan	Detail Edit Hapus
10	88888888	88888888	Administrator	Detail Edit Hapus

- Menampilkan data yang baru ditambahkan tanpa me reload web

## Praktikum 2. Modal Ajax Edit Data (Data User)

1. Pada Praktikum 2 ini, kita akan melakukan koding untuk proses edit menggunakan ajax.
2. Pertama-tama, kita **ubah** dulu fungsi `list()` pada `UserController.php` untuk mengganti **tombol edit** untuk bisa menggunakan modal

```
// Ambil data user dalam bentuk json untuk datatables
public function list(Request $request)
{
    $users = UserModel::select('user_id', 'username', 'nama', 'level_id')
        ->with('level');

    // Filter data user berdasarkan level_id
    if ($request->level_id){
        $users->where('level_id',$request->level_id);
    }

    return DataTables::of($users)
        ->addIndexColumn() // menambahkan kolom index / no urut (default nama kolom: DT_RowIndex)
        ->addColumn('aksi', function ($user) { // menambahkan kolom aksi
```



```
/* $btn = '<a href="'.url('/user/' . $user->user_id).'>Detail</a>' class="btn btn-info btn-sm">Detail</a>';  
$btn .= '<a href="'.url('/user/' . $user->user_id . '/edit').">Edit</a>' class="btn btn-warning btn-sm">Edit</a>';  
$btn .= '<form class="d-inline-block" method="POST" action="'.url('/user/' . $user->user_id).'>  
    . csrf_field() . method_field('DELETE') .  
    '<button type="submit" class="btn btn-danger btn-sm" onclick="return  
confirm(\''Apakah Anda yakin menghapus data ini?'\');">Hapus</button></form>';*/  
$btn = '<button onclick="modalAction(\'' . url('/user/' . $user->user_id .  
'/show_ajax') . '\')>Detail</button>' class="btn btn-info btn-sm">Detail</button>';  
$btn .= '<button onclick="modalAction(\'' . url('/user/' . $user->user_id .  
'/edit_ajax') . '\')>Edit</button>' class="btn btn-warning btn-sm">Edit</button>';  
$btn .= '<button onclick="modalAction(\'' . url('/user/' . $user->user_id .  
'/delete_ajax') . '\')>Hapus</button>' class="btn btn-danger btn-sm">Hapus</button>';  
  
return $btn;  
})  
->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html  
->make(true);  
}
```

#### UserController function list

```
40 public function list(Request $request)  
41 {  
42     $users = UserModel::select('user_id', 'username', 'nama', 'level_id')  
43     ->with('level');  
44  
45     // Filter data user berdasarkan level_id  
46     if ($request->level_id) {  
47         $users->where('level_id', $request->level_id);  
48     }  
49  
50     return DataTables::of($users)  
51     ->addIndexColumn() // Menambahkan kolom index / no urut (default nama kolom: DT_RowIndex)  
52     ->addColumn('aksi', function ($user) {  
53         /*  
54         $btn = '<a href="'.url('/user/' . $user->user_id).'>Detail</a>' class="btn btn-info btn-sm">Detail</a>';  
55         $btn .= '<a href="'.url('/user/' . $user->user_id . '/edit').">Edit</a>' class="btn btn-warning btn-sm">Edit</a>';  
56         $btn .= '<form class="d-inline-block" method="POST" action="'.url('/user/' . $user->user_id).'>  
57         . csrf_field() . method_field('DELETE') .  
58         '<button type="submit" class="btn btn-danger btn-sm" onclick="return confirm(\''Apakah Anda yakin menghapus data ini?'\');">Hapus</button></form>';  
59         */  
60  
61         $btn = '<button onclick="modalAction(\'' . url('/user/' . $user->user_id . '/show_ajax') . '\')>Detail</button>';  
62         $btn .= '<button onclick="modalAction(\'' . url('/user/' . $user->user_id . '/edit_ajax') . '\')>Edit</button>';  
63         $btn .= '<button onclick="modalAction(\'' . url('/user/' . $user->user_id . '/delete_ajax') . '\')>Hapus</button>';  
64  
65         return $btn;  
66     });  
67     ->rawColumns(['aksi']) // Memberitahu bahwa kolom aksi adalah HTML  
68     ->make(true);  
69 }
```

3. Selanjutnya kita modifikasi `routes/web.php` untuk mengakomodir request edit menggunakan ajax

```
Route::group(['prefix' => 'user'], function () {  
    Route::get('/', [UserController::class, 'index']); // Menampilkan halaman awal user  
    Route::post('/list', [UserController::class, 'list']); // Menampilkan data user dalam bentuk json untuk datatable  
    Route::get('/create', [UserController::class, 'create']); // Menampilkan halaman form tambah user  
    Route::post('/', [UserController::class, 'store']); // Menyimpan data user baru  
    Route::post('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax  
    Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax  
    Route::get('/{id}', [UserController::class, 'show']); // Menampilkan detail user  
    Route::get('/{id}/edit', [UserController::class, 'edit']); // Menampilkan halaman form edit user  
    Route::put('/{id}', [UserController::class, 'update']); // Menyimpan perubahan data user  
    Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax  
    Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax  
    Route::delete('/{id}', [UserController::class, 'destroy']); // Menghapus data user  
});
```

```
52 Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // menampilkan halaman form edit user ajax  
53 Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // menyimpan perubahan data user ajax
```



4. Kemudian, kita buat fungsi `edit_ajax()` pada `UserController.php`

```
// Menampilkan halaman form edit user ajax
public function edit_ajax(string $id)
{
    $user = UserModel::find($id);
    $level = LevelModel::select('level_id', 'level_nama')->get();

    return view('user.edit_ajax', ['user' => $user, 'level' => $level]);
}
```

```
211 public function edit_ajax(string $id){
212     $user = UserModel::find($id);
213     $level = LevelModel::select('level_id', 'level_nama')->get();
214
215     return view('user.edit_ajax', ['user'=> $user, 'level' => $level]);
216 }
217
```

5. Kita buat **view baru** pada `user/edit_ajax.blade.php` untuk menampilkan form view ajax

```
@empty($user)
<div id="modal-master" class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
        <div class="modal-header">
            <h5 class="modal-title" id="exampleModallabel">Kesalahan</h5>
            <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
```





```
</div>
<div class="modal-body">
  <div class="alert alert-danger">
    <h5><i class="icon fas fa-ban"></i> Kesalahan!!!</h5>
    Data yang anda cari tidak ditemukan</div>
    <a href="{{ url('/user') }}" class="btn btn-warning">Kembali</a>
  </div>
</div>
</div>
@else
  <form action="{{ url('/user/' . $user->user_id.'/update_ajax') }}" method="POST" id="form-
edit">
    @csrf
    @method('PUT')
    <div id="modal-master" class="modal-dialog modal-lg" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Edit Data User</h5>
          <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
        </div>
        <div class="modal-body">
          <div class="form-group">
            <label>Level Pengguna</label>
            <select name="level_id" id="level_id" class="form-control" required>
              <option value="">- Pilih Level -</option>
              @foreach($level as $l)
                <option {{ ($l->level_id == $user->level_id)? 'selected' : '' }}
value="{{ $l->level_id }}">{{ $l->level_nama }}</option>
              @endforeach
            </select>
            <small id="error-level_id" class="error-text form-text text-
danger"></small>
          </div>
          <div class="form-group">
            <label>Username</label>
            <input value="{{ $user->username }}" type="text" name="username"
id="username" class="form-control" required>
            <small id="error-username" class="error-text form-text text-
danger"></small>
          </div>
          <div class="form-group">
            <label>Nama</label>
            <input value="{{ $user->nama }}" type="text" name="nama" id="nama"
class="form-control" required>
            <small id="error-nama" class="error-text form-text text-danger"></small>
          </div>
          <div class="form-group">
            <label>Password</label>
            <input value="" type="password" name="password" id="password" class="form-
control">
            <small class="form-text text-muted">Abaikan jika tidak ingin ubah
password</small>
            <small id="error-password" class="error-text form-text text-
danger"></small>
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" data-dismiss="modal" class="btn btn-
warning">Batal</button>
          <button type="submit" class="btn btn-primary">Simpan</button>
        </div>
      </div>
    </div>
  </form>
</script>
```





```
$(document).ready(function() {
    $("#form-edit").validate({
        rules: {
            level_id: {required: true, number: true},
            username: {required: true, minlength: 3, maxlength: 20},
            nama: {required: true, minlength: 3, maxlength: 100},
            password: {minlength: 6, maxlength: 20}
        },
        submitHandler: function(form) {
            $.ajax({
                url: form.action,
                type: form.method,
                data: $(form).serialize(),
                success: function(response) {
                    if(response.status){
                        $('#myModal').modal('hide');
                        Swal.fire({
                            icon: 'success',
                            title: 'Berhasil',
                            text: response.message
                        });
                        dataUser.ajax.reload();
                    }else{
                        $('.error-text').text('');
                        $.each(response.msgField, function(prefix, val) {
                            $('#error-'+prefix).text(val[0]);
                        });
                        Swal.fire({
                            icon: 'error',
                            title: 'Terjadi Kesalahan',
                            text: response.message
                        });
                    }
                }
            });
            return false;
        },
        errorElement: 'span',
        errorPlacement: function (error, element) {
            error.addClass('invalid-feedback');
            element.closest('.form-group').append(error);
        },
        highlight: function (element, errorClass, validClass) {
            $(element).addClass('is-invalid');
        },
        unhighlight: function (element, errorClass, validClass) {
            $(element).removeClass('is-invalid');
        }
    });
});
</script>
@endempty
```



```
as > user > edit_ajax.blade.php > form#form-edit > div#modal-master.modal-dialog.modal-lg > div.modal-content > div.modal-body > div.form-group > small#error-password.en
1 @empty($user)
2     <div id="modal-master" class="modal-dialog modal-lg" role="document">
3         <div class="modal-content">
4             <div class="modal-header">
5                 <h5 class="modal-title" id="exampleModallabel">Kesalahan</h5>
6                 <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
7                     aria-hidden="true">&times;</span></button>
8             </div>
9             <div class="modal-body">
10                <div class="alert alert-danger">
11                    <h5><i class="icon fas fa-ban"></i> Kesalahan!!!</h5> Data yang anda cari tidak ditemukan
12                </div>
13                <a href="{{ url('/user') }}" class="btn btn-warning">Kembali</a>
14            </div>
15        </div>
16    </div>
17 @else
18     <form action="{{ url('/user/' . $user->user_id . '/update_ajax') }}" method="POST" id="form-edit">
19         @csrf @method('PUT')
20         <div id="modal-master" class="modal-dialog modal-lg" role="document">
21             <div class="modal-content">
22                 <div class="modal-header">
23                     <h5 class="modal-title" id="exampleModallabel">Edit Data User</h5>
24                     <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
25                         aria-hidden="true">&times;</span></button>
26                 </div>
27                 <div class="modal-body">
28                     <div class="form-group">
29                         <label>Level Pengguna</label>
30                         <select name="level_id" id="level_id" class="form-control" required>
31                             <option value="">- Pilih Level -</option>
32                             @foreach ($level as $l)
33                                 <option {{ $l->level_id == $user->level_id ? 'selected' : '' }} value="{{ $l->level_id }}">
34                                     {{ $l->level_nama }}</option>
35                             @endforeach
36                         </select>
37                         <small id="error-level_id" class="error-text form-text text-danger"></small>

```

6. Selanjutnya, kita buat fungsi `update_ajax()` pada `UserController.php` untuk mengakomodir request update data user melalui ajax

```
public function update_ajax(Request $request, $id){
    // cek apakah request dari ajax
    if ($request->ajax() || $request->wantsJson()) {
        $rules = [
            'level_id' => 'required|integer',
            'username' => 'required|max:20|unique:m_user,username, '.$id.',user_id',
            'nama' => 'required|max:100',
            'password' => 'nullable|min:6|max:20'
        ];
    }
}
```



```
// use Illuminate\Support\Facades\Validator;
$validator = Validator::make($request->all(), $rules);

if ($validator->fails()) {
    return response()->json([
        'status' => false,    // respon json, true: berhasil, false: gagal
        'message' => 'Validasi gagal.',
        'msgField' => $validator->errors() // menunjukkan field mana yang error
    ]);
}

$check = UserModel::find($id);
if ($check) {
    if (!$request->filled('password')) { // jika password tidak diisi, maka hapus dari
request
        $request->request->remove('password');
    }

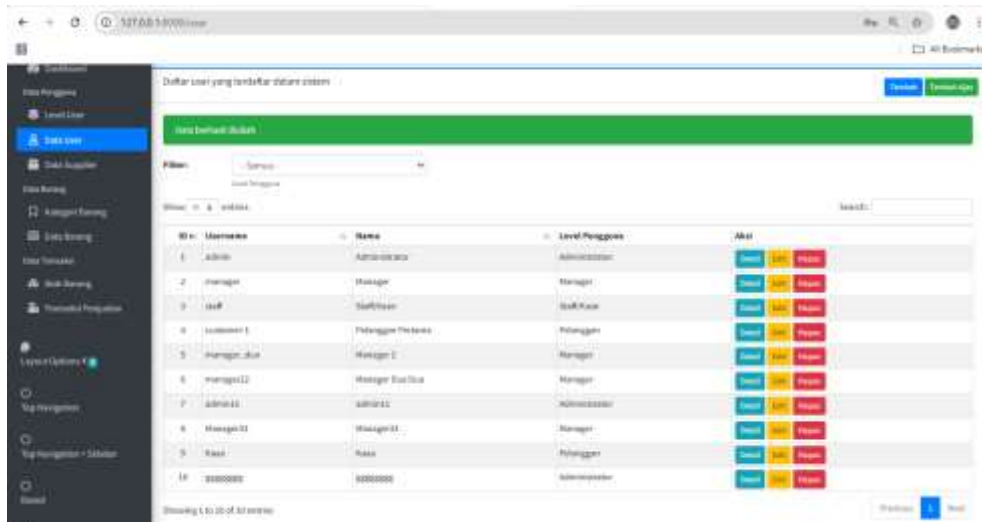
    $check->update($request->all());
    return response()->json([
        'status' => true,
        'message' => 'Data berhasil diupdate'
    ]);
} else {
    return response()->json([
        'status' => false,
        'message' => 'Data tidak ditemukan'
    ]);
}
}
return redirect('/');
}
```

```
182 public function store_ajax(Request $request){
183     //cek req ajax
184     if ($request->ajax() || $request->wantsJson()) {
185         $rules = [
186             'level_id' => 'required|integer',
187             'username' => 'required|string|min:3|unique:m_user,username',
188             'nama' => 'required|string|max:100',
189             'password' => 'required|min:6',
190         ];
191     };
192
193     $validator = Validator::make($request->all(), $rules);
194
195     if ($validator->fails()) {
196         return response()->json([
197             'status' => false, //response status
198             'message' => 'Validasi Gagal', //pesan gagal
199             'msgField' => $validator->errors(), //pesan error
200         ]);
201     }
202     UserModel::create($request->all());
203     return response()->json([
204         'status' => true,
205         'message' => 'data user berhasil disimpan'
206     ]);
207 }
208 redirect('/');
209 }
210
211 public function edit_ajax(string $id){
212     $user = UserModel::find($id);
213     $level = LevelModel::select('level_id', 'level_nama')->get();
214
215     return view('user.edit_ajax', ['user' => $user, 'level' => $level]);
216 }
```

7. Sekarang kita coba bagian edit user, amati proses nya. Jangan lupa laporkan dan *commit*



ke repository git kalian !



### Praktikum 3. Modal Ajax Hapus Data (Data User)

1. Pada Praktikum 3 ini, kita akan melakukan koding untuk proses hapus menggunakan ajax.
2. Pertama-tama, kita ubah dulu `routes/web.php` untuk mengakomodir request halaman konfirmasi untuk menghapus data

```
Route::group(['prefix' => 'user'], function () {  
    Route::get('/', [UserController::class, 'index']); // Menampilkan halaman awal user  
    Route::post('/list', [UserController::class, 'list']); // Menampilkan data user dalam bentuk json untuk datatables  
    Route::get('/create', [UserController::class, 'create']); // Menampilkan halaman form tambah user  
    Route::post('/', [UserController::class, 'store']); // Menyimpan data user baru  
    Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax  
    Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax  
    Route::get('/{id}', [UserController::class, 'show']); // Menampilkan detail user  
    Route::get('/{id}/edit', [UserController::class, 'edit']); // Menampilkan halaman form edit user  
    Route::put('/{id}', [UserController::class, 'update']); // Menyimpan perubahan data user  
    Route::get('/{id}/edit_ajax', [UserController::class, 'edit_ajax']); // Menampilkan halaman form edit user Ajax  
    Route::put('/{id}/update_ajax', [UserController::class, 'update_ajax']); // Menyimpan perubahan data user Ajax  
    Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); // Untuk tampilan form confirm delete user Ajax  
    Route::delete('/{id}/delete_ajax', [UserController::class, 'delete_ajax']); // Untuk hapus data user Ajax  
    Route::delete('/{id}', [UserController::class, 'destroy']); // Menghapus data user  
});
```

```
54 Route::get('/{id}/delete_ajax',[UserController::class, 'confirm_ajax']); // untuk tampilan form confirm delete ajax js 6  
55 Route::delete('/{id}/delete_ajax',[UserController::class, 'delete_ajax']); // untuk hapus data ajax js 6
```

3. Kemudian kita buat fungsi `confirm_ajax()` pada `UserController.php`



```
public function confirm_ajax(string $id){  
    $user = UserModel::find($id);  
  
    return view('user.confirm_ajax', ['user' => $user]);  
}
```

```
264  
265     public function confirm_ajax(string $id){  
266         $user = UserModel::find($id);  
267  
268         return view('user.confirm_ajax', ['user' => $user]);  
269     }  
270
```

4. Selanjutnya kita view untuk konfirmasi hapus data dengan nama `user/confirm_ajax.blade.php`

```
@empty($user)  
    <div id="modal-master" class="modal-dialog modal-lg" role="document">  
        <div class="modal-content">  
            <div class="modal-header">  
                <h5 class="modal-title" id="exampleModallabel">Kesalahan</h5>  
                <button type="button" class="close" data-dismiss="modal" aria-  
label="Close"><span aria-hidden="true">&times;</span></button>  
            </div>  
            <div class="modal-body">  
                <div class="alert alert-danger">  
                    <h5><i class="icon fas fa-ban"></i> Kesalahan!!!</h5>  
                    Data yang anda cari tidak ditemukan</div>  
                    <a href="{{ url('/user') }}" class="btn btn-warning">Kembali</a>  
                </div>  
            </div>  
        </div>  
    @else  
        <form action="{{ url('/user/' . $user->user_id . '/delete_ajax') }}" method="POST" id="form-  
delete">  
            @csrf  
            @method('DELETE')  
            <div id="modal-master" class="modal-dialog modal-lg" role="document">  
                <div class="modal-content">  
                    <div class="modal-header">  
                        <h5 class="modal-title" id="exampleModallabel">Hapus Data User</h5>  
                        <button type="button" class="close" data-dismiss="modal" aria-  
label="Close"><span aria-hidden="true">&times;</span></button>  
                    </div>  
                    <div class="modal-body">  
                        <div class="alert alert-warning">  
                            <h5><i class="icon fas fa-ban"></i> Konfirmasi !!!</h5>  
                            Apakah Anda ingin menghapus data seperti di bawah ini?  
                        </div>  
                        <table class="table table-sm table-bordered table-striped">  
                            <tr><th class="text-right col-3">Level Pengguna :</th><td class="col-9">{{  
$user->level->level_nama }}</td></tr>  
                            <tr><th class="text-right col-3">Username :</th><td class="col-9">{{  
$user->username }}</td></tr>  
                            <tr><th class="text-right col-3">Nama :</th><td class="col-9">{{ $user-&br/>>nama }}</td></tr>  
                        </table>  
                    </div>  
                    <div class="modal-footer">  
                        <button type="button" data-dismiss="modal" class="btn btn-  
warning">Batal</button>  
                        <button type="submit" class="btn btn-primary">Ya, Hapus</button>  
                    </div>  
                </div>  
            </div>  
        </form>  
    </script>
```



```
url: form.action,
type: form.method,
data: $(form).serialize(),
success: function(response) {
    if(response.status){
        $('#myModal').modal('hide');
        Swal.fire({
            icon: 'success',
            title: 'Berhasil',
            text: response.message
        });
        dataUser.ajax.reload();
    }else{
        $('.error-text').text('');
        $.each(response.msgField, function(prefix, val) {
            $('#error-'+prefix).text(val[0]);
        });
        Swal.fire({
            icon: 'error',
            title: 'Terjadi Kesalahan',
            text: response.message
        });
    }
}
});
return false;
},
errorElement: 'span',
errorPlacement: function (error, element) {
    error.addClass('invalid-feedback');
    element.closest('.form-group').append(error);
},
highlight: function (element, errorClass, validClass) {
    $(element).addClass('is-invalid');
},
unhighlight: function (element, errorClass, validClass) {
    $(element).removeClass('is-invalid');
}
});
});
</script>
@endempty
```

5. Kemudian kita buat fungsi `delete_ajax()` pada `UserController.php` untuk mengakomodir *request* hapus data user



```
Minggu_6 > PWL_POS > resources > views > user > confirm_ajax.blade.php > form#form-delete > div#modal-master.modal-dialog.modal-lg > div.modal-conte

1  @empty($user)
2  <div id="modal-master" class="modal-dialog modal-lg" role="document">
3    <div class="modal-content">
4      <div class="modal-header">
5        <h5 class="modal-title" id="exampleModallabel">Kesalahan</h5>
6        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
7          <span aria-hidden="true">&times;</span>
8        </button>
9      </div>
10     <div class="modal-body">
11       <div class="alert alert-danger">
12         <h5><i class="icon fas fa-ban"></i> Kesalahan!!!</h5>
13         Data yang anda cari tidak ditemukan
14       </div>
15       <a href="{{ url('/user') }}" class="btn btn-warning">Kembali</a>
16     </div>
17   </div>
18 </div>
19 @else
20 <form action="{{ url('/user/' . $user->user_id . '/delete_ajax') }}" method="POST" id="form-delete">
21   @csrf
22   @method('DELETE')
23   <div id="modal-master" class="modal-dialog modal-lg" role="document">
24     <div class="modal-content">
25       <div class="modal-header">
26         <h5 class="modal-title" id="exampleModallabel">Hapus Data User</h5>
27         <button type="button" class="close" data-dismiss="modal" aria-label="Close">
28           <span aria-hidden="true">&times;</span>
29         </button>
30       </div>
31       <div class="modal-body">
32         <div class="alert alert-warning">
33           <h5><i class="icon fas fa-ban"></i> Konfirmasi !!!</h5>
34           Apakah Anda ingin menghapus data seperti di bawah ini?
35         </div>
36         <table class="table table-sm table-bordered table-striped">
```





```
public function delete_ajax(Request $request, $id)
{
    // cek apakah request dari ajax
    if ($request->ajax() || $request->wantsJson()) {
        $user = UserModel::find($id);
        if ($user) {
            $user->delete();
            return response()->json([
                'status' => true,
                'message' => 'Data berhasil dihapus'
            ]);
        } else {
            return response()->json([
                'status' => false,
                'message' => 'Data tidak ditemukan'
            ]);
        }
    }
    return redirect('/');
}
```

```
271 public function delete_ajax(Request $request, $id){
272     if ($request->ajax() || $request->wantsJson()) {
273         $user = UserModel::find($id);
274         if ($user) {
275             $user->delete();
276             return response()->json([
277                 'status' => true,
278                 'message' => 'data berhasil dihapus'
279             ]);
280         } else {
281             return response()->json([
282                 'status' => false,
283                 'message' => 'data tidak ditemukan'
284             ]);
285         }
286     }
287 }
```

6. Setelah semua selesai, mari kita coba untuk melakukan percobaan dari koding yang telah kita lakukan.





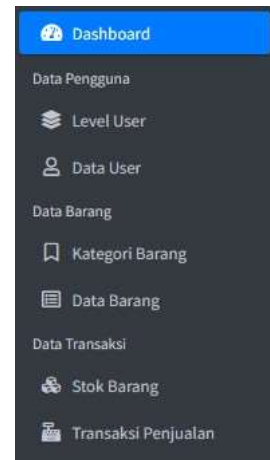
7. Jangan lupa laporkan ke laporan jobsheet dan lakukan *commit* pada *repository git* kalian.!!!

## F. Tugas Jobsheet

Implementasikan koding untuk Ajax Form dan Client Validation dengan jQuery Validation pada menu berikut ini

- ✓ Tabel m\_level
- ✓ Tabel m\_kategori
- ✓ Tabel m\_supplier
- ✓ Tabel m\_barang

Laporkan pada laporan jobsheet dan Jangan lupa di commit dan push pada repository git kalian.



Tugas

M\_level

- LevelController

Create\_ajax

```
141 public function create_ajax(){
142     return view('level.create_ajax');
143 }
```

Store\_ajax

```
145 public function store_ajax(Request $request){
146     //cek req ajax
147     if ($request->ajax() || $request->wantsJson()) {
148         $rules = [
149             'level_kode' => 'required|string|unique:m_level,level_kode',
150             'level_nama' => 'required|string',
151         ];
152
153         $validator = Validator::make($request->all(), $rules);
154
155         if ($validator->fails()) {
156             return response()->json([
157                 'status' => false, //response status
158                 'message' => 'Validasi Gagal', //pesan gagal
159                 'msgField' => $validator->errors(), //pesan error
160             ]);
161         }
162         LevelModel::create($request->all());
163         return response()->json([
164             'status' => true,
165             'message' => 'data user berhasil disimpan'
166         ]);
167     }
168     redirect('/');
169 }
```



#### Edit\_ajax

```
176 public function edit_ajax(string $id){  
177     $level = LevelModel::find($id);  
178     return view('level.edit_ajax', ['level' => $level]);  
179 }
```

#### Update\_ajax

```
181 public function update_ajax(Request $request, $id){  
182     // Cek apakah request berasal dari AJAX  
183     if ($request->ajax() || $request->wantsJson()) {  
184         $rules = [  
185             'level_kode' => 'required|string|unique:m_level,level_kode, '.$id.',level_id',  
186             'level_nama' => 'required|string',  
187         ];  
188  
189         // Validasi input  
190         $validator = Validator::make($request->all(), $rules);  
191  
192         if ($validator->fails()) {  
193             return response()->json([  
194                 'status' => false, // Respon JSON, true: berhasil, false: gagal  
195                 'message' => 'Validasi gagal.',  
196                 'msgField' => $validator->errors() // Menunjukkan field mana yang error  
197             ]);  
198         }  
199  
200         // Cek apakah user dengan ID yang diberikan ada  
201         $check = LevelModel::find($id);  
202         if ($check) {  
203  
204             $check->update($request->all());  
205  
206             return response()->json([  
207                 'status' => true,  
208                 'message' => 'Data berhasil diupdate'  
209             ]);  
210         } else {  
211             return response()->json([  
212                 'status' => false,  
213                 'message' => 'Data tidak ditemukan'  
214             ]);  
215         }  
216     }  
217 }
```

#### Confirm\_ajax

```
221 public function confirm_ajax(string $id){  
222     $level = LevelModel::find($id);  
223     return view('level.confirm_ajax', ['level' => $level]);  
224 }
```

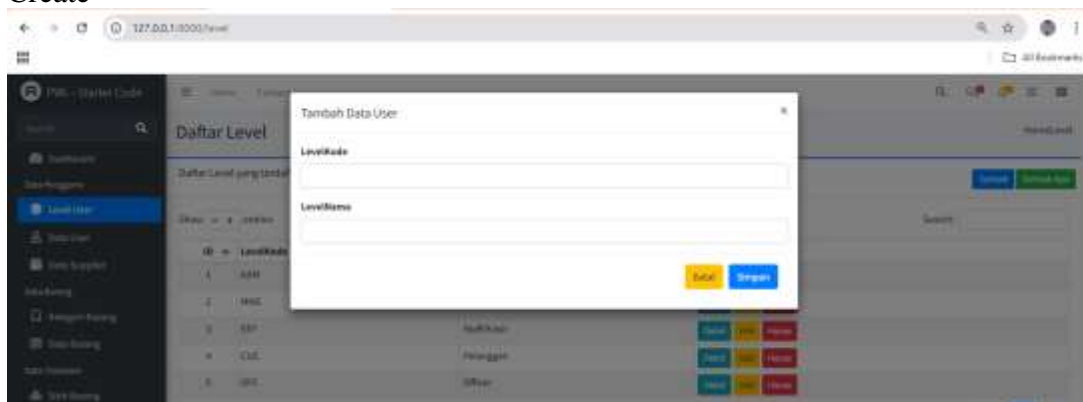


### Delete\_ajax

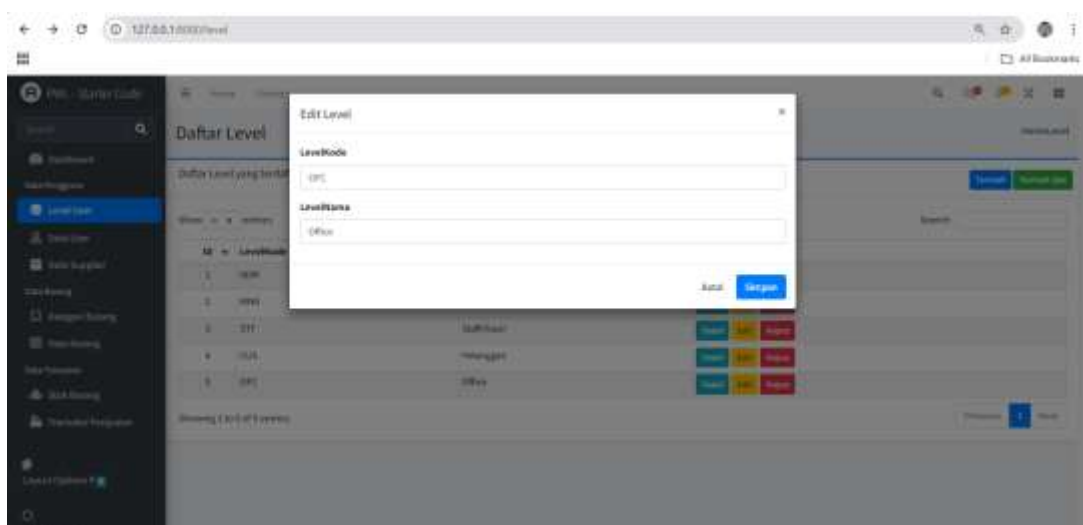
```
226 public function delete_ajax(Request $request, $id){  
227     if ($request->ajax() || $request->wantsJson()) {  
228         $user = LevelModel::find($id);  
229         if ($user) {  
230             $user->delete();  
231             return response()->json([  
232                 'status' => true,  
233                 'message' => 'data berhasil dihapus'  
234             ]);  
235         } else {  
236             return response()->json([  
237                 'status' => false,  
238                 'message' => 'data tidak ditemukan'  
239             ]);  
240         }  
241     }  
}
```

- View

### Create



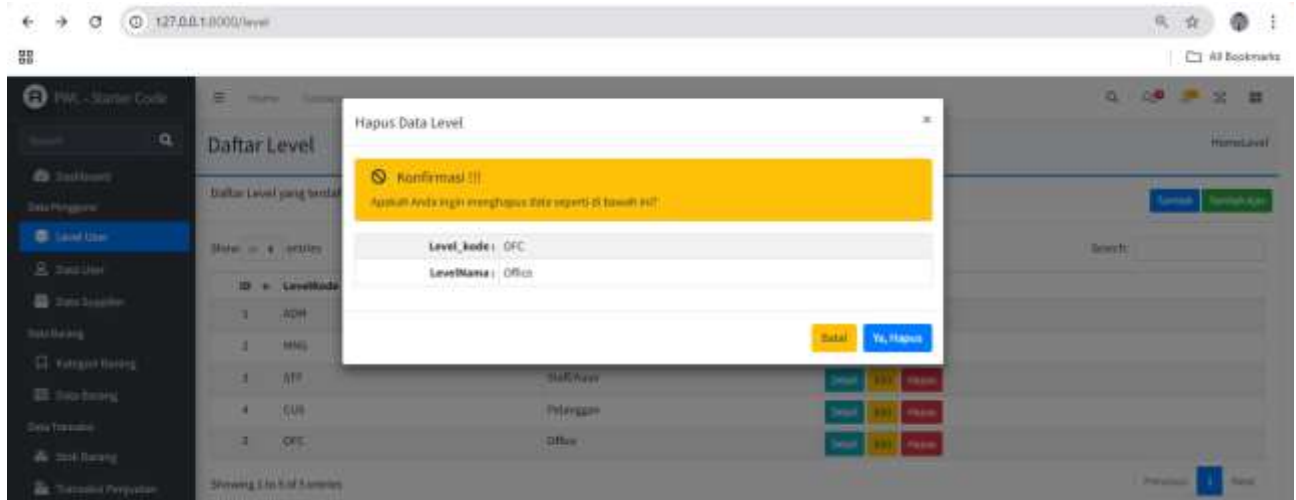
### Edit





Delete

tor



M\_kategori

- KategoriController

Create\_ajax

```
140 public function create_ajax(){
141     return view('kategori.create_ajax');
142 }
143
```

Store\_ajax

```
144 public function store_ajax(Request $request){
145     //cek req ajax
146     if ($request->ajax() || $request->wantsJson()) {
147         $rules = [
148             'kategori_kode' => 'required|string|unique:m_kategori,kategori_kode',
149             'kategori_nama' => 'required|string',
150         ];
151
152         $validator = Validator::make($request->all(), $rules);
153
154         if ($validator->fails()) {
155             return response()->json([
156                 'status' => false, //response status
157                 'message' => 'Validasi Gagal', //pesan gagal
158                 'msgField' => $validator->errors(), //pesan error
159             ]);
160         }
161         KategoriModel::create($request->all());
162         return response()->json([
163             'status' => true,
164             'message' => 'data user berhasil disimpan'
165         ]);
166     }
167     redirect('/');
168 }
169
```



### Edit\_ajax

```
170 public function edit_ajax(string $id){
171     $kategori = KategoriModel::find($id);
172     return view('kategori.edit_ajax', ['kategori' => $kategori]);
173 }
174
```

### Update\_ajax

```
175 public function update_ajax(Request $request, $id){
176     if ($request->ajax() || $request->wantsJson()) {
177         $rules = [
178             'kategori_kode' => 'required|string|unique:m_kategori,kategori_kode,.'. $id.',kategori_id',
179             'kategori_nama' => 'required|string',
180         ];
181     };
182
183     // Validasi input
184     $validator = Validator::make($request->all(), $rules);
185
186     if ($validator->fails()) {
187         return response()->json([
188             'status' => false, // Respon JSON, true: berhasil, false: gagal
189             'message' => 'Validasi gagal.',
190             'msgField' => $validator->errors() // Menunjukkan field mana yang error
191         ]);
192     }
193
194     // Cek apakah user dengan ID yang diberikan ada
195     $check = KategoriModel::find($id);
196     if ($check) {
197         $check->update($request->all());
198
199         return response()->json([
200             'status' => true,
201             'message' => 'Data berhasil diupdate'
202         ]);
203     } else {
204         return response()->json([
205             'status' => false,
206             'message' => 'Data tidak ditemukan'
207         ]);
208     }
209 }
210
211
212 return redirect('/');
```

### Confirm\_ajax

```
215 public function confirm_ajax(string $id){
216     $kategori = KategoriModel::find($id);
217     return view('kategori.confirm_ajax', ['kategori' => $kategori]);
218 }
219
```

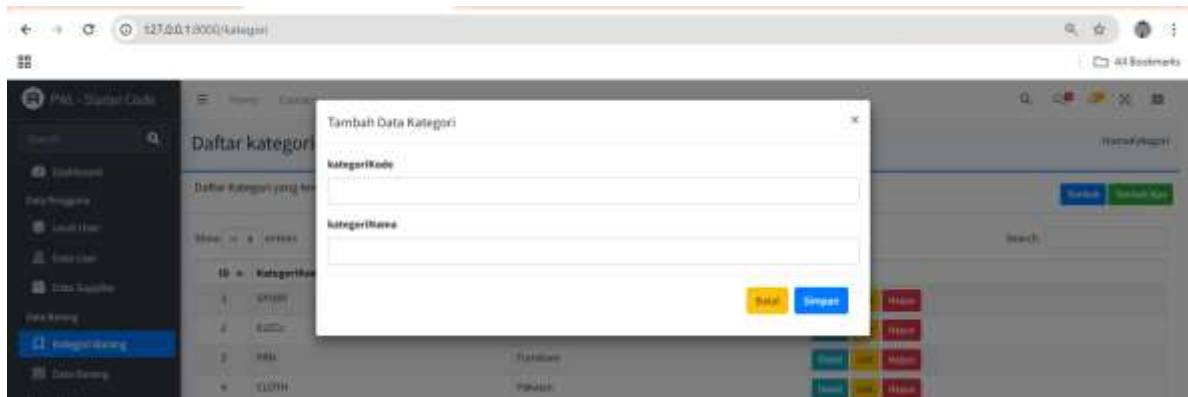
### Delete\_ajax

```
220 public function delete_ajax(Request $request, $id){
221     if ($request->ajax() || $request->wantsJson()) {
222         $user = KategoriModel::find($id);
223         if ($user) {
224             $user->delete();
225             return response()->json([
226                 'status' => true,
227                 'message' => 'data berhasil dihapus'
228             ]);
229         } else {
230             return response()->json([
231                 'status' => false,
232                 'message' => 'data tidak ditemukan'
233             ]);
234         }
235     }
236 }
```



- View

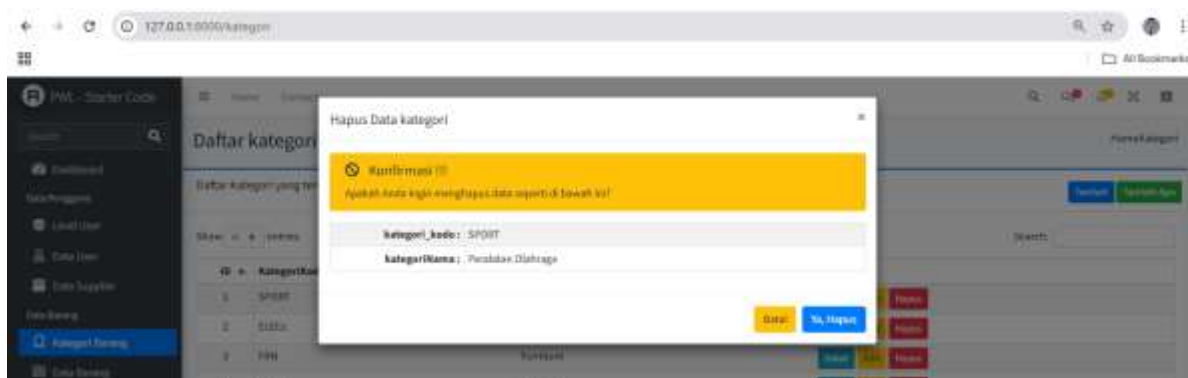
## Create



## Edit



## Delete







M\_barang

- BarangController

Create\_ajax

```
155 public function create_ajax(){
156     $kategori = KategoriModel::select('kategori_id','kategori_nama')->get();
157
158     return View('barang.create_ajax')
159     ->with('kategori', $kategori);
160 }
```

Store\_ajax

```
162 public function store_ajax(Request $request){
163     //cek req ajax
164     if ($request->ajax() || $request->wantsJson()) {
165         $rules = [
166             'barang_kode' => 'required|string|max:100',
167             'barang_nama' => 'required|min:5',
168             'harga_beli' => 'required|integer',
169             'harga_jual' => 'required|integer',
170             'kategori_id' => 'required|integer',
171         ];
172
173         $validator = Validator::make($request->all(), $rules);
174
175         if ($validator->fails()) {
176             return response()->json([
177                 'status' => false, //response status
178                 'message' => 'Validasi Gagal', //pesan gagal
179                 'msgField' => $validator->errors(), //pesan error
180             ]);
181         }
182         BarangModel::create($request->all());
183         return response()->json([
184             'status' => true,
185             'message' => 'data user berhasil disimpan'
186         ]);
187     }
188     redirect('/');
189 }
```

Edit\_ajax

```
191 public function edit_ajax(string $id){
192     $barang = BarangModel::find($id);
193     $kategori = KategoriModel::select('kategori_id', 'kategori_nama')->get();
194
195     return view('barang.edit_ajax', ['barang' => $barang, 'kategori' => $kategori]);
196 }
```



## Update\_ajax

```
198 public function update_ajax(Request $request, $id){
199     if ($request->ajax() || $request->wantsJson()) {
200         $rules = [
201             'barang_kode' => 'required|string|max:100',
202             'barang_nama' => 'required|min:5',
203             'harga_beli' => 'required|integer',
204             'harga_jual' => 'required|integer',
205             'kategori_id' => 'required|integer',
206         ];
207     };
208
209     // Validasi input
210     $validator = Validator::make($request->all(), $rules);
211
212     if ($validator->fails()) {
213         return response()->json([
214             'status' => false, // Respon JSON, true: berhasil, false: gagal
215             'message' => 'Validasi gagal.',
216             'msgField' => $validator->errors() // Menunjukkan field mana yang error
217         ]);
218     }
219
220     // Cek apakah user dengan ID yang diberikan ada
221     $check = BarangModel::find($id);
222     if ($check) {
223         // Jika password tidak diisi, hapus dari request
224         if (!$request->filled('password')) {
225             $request->request->remove('password');
226         }
227
228         // Update data user
229         $check->update($request->all());
230
231         return response()->json([
232             'status' => true,
233             'message' => 'Data berhasil diupdate'
234         ]);
235     } else {
236         return response()->json([
237             'status' => false,
238             'message' => 'Data tidak ditemukan'
239         ]);
240     }
241 }
```

## Confirm\_ajax

```
245 public function confirm_ajax(string $id){
246     $barang = BarangModel::find($id);
247
248     return view('barang.confirm_ajax', ['barang' => $barang]);
249 }
```

## Delete\_ajax

```
251 public function delete_ajax(Request $request, $id){
252     if ($request->ajax() || $request->wantsJson()) {
253         $barang = BarangModel::find($id);
254         if ($barang) {
255             $barang->delete();
256             return response()->json([
257                 'status' => true,
258                 'message' => 'data berhasil dihapus'
259             ]);
260         } else {
261             return response()->json([
262                 'status' => false,
263                 'message' => 'data tidak ditemukan'
264             ]);
265         }
266     }
267 }
```



- View

## Create

Tambah Data Barang

Kategori Barang:

BarangKode:

BarangNama:

HargaBeli:

HargaJual:

## Edit

Edit Data Barang

Kategori Barang:

BarangKode:

BarangNama:

HargaBeli:

HargaJual:

## Delete

Hapus Data Barang

Konfirmasi !!  
Apakah Anda ingin menghapus data seperti di bawah ini?

Kategori Barang: Peralatan Olahraga

BarangKode: 8000

BarangNama: Sepeda Road

HargaBeli: 1500000

HargaJual: 2000000



M\_supplier

- SupplierController

Create\_ajax

```
139  
140     public function create_ajax(){  
141         return view('supplier.create_ajax');  
142     }  
143
```

Store\_ajax

```
144     public function store_ajax(Request $request){  
145         //cek req ajax  
146         if ($request->ajax() || $request->wantsJson()) {  
147             $rules = [  
148                 'supplier_kode' => 'required|string|unique:m_supplier,supplier_kode',  
149                 'supplier_nama' => 'required|string',  
150             ];  
151  
152             $validator = Validator::make($request->all(), $rules);  
153  
154             if ($validator->fails()) {  
155                 return response()->json([  
156                     'status' => false, //response status  
157                     'message' => 'Validasi Gagal', //pesan gagal  
158                     'msgField' => $validator->errors(), //pesan error  
159                 ]);  
160             }  
161             SupplierModel::create($request->all());  
162             return response()->json([  
163                 'status' => true,  
164                 'message' => 'data user berhasil disimpan'  
165             ]);  
166         }  
167         redirect('/');  
168     }
```

Edit\_ajax

```
170     public function edit_ajax(string $id){  
171         $supplier = SupplierModel::find($id);  
172         return view('supplier.edit_ajax',['supplier' => $supplier]);  
173     }
```



## Update\_ajax

```
175 public function update_ajax(Request $request, $id){
176     // Cek apakah request berasal dari AJAX
177     if ($request->ajax() || $request->wantsJson()) {
178         $rules = [
179             'supplier_kode' => 'required|string|unique:m_supplier,supplier_kode, '.$id.',supplier_id',
180             'supplier_nama' => 'required|string',
181         ];
182
183         // Validasi input
184         $validator = Validator::make($request->all(), $rules);
185
186         if ($validator->fails()) {
187             return response()->json([
188                 'status' => false, // Respon JSON, true: berhasil, false: gagal
189                 'message' => 'Validasi gagal.',
190                 'msgField' => $validator->errors() // Menunjukkan field mana yang error
191             ]);
192         }
193
194         // Cek apakah user dengan ID yang diberikan ada
195         $check = SupplierModel::find($id);
196         if ($check) {
197             $check->update($request->all());
198
199             return response()->json([
200                 'status' => true,
201                 'message' => 'Data berhasil diupdate'
202             ]);
203         } else {
204             return response()->json([
205                 'status' => false,
206                 'message' => 'Data tidak ditemukan'
207             ]);
208         }
209     }
```

## Confirm\_ajax

```
215 public function confirm_ajax(string $id){
216     $supplier = SupplierModel::find($id);
217     return view('supplier.confirm_ajax', ['supplier' => $supplier]);
218 }
```

## Delete\_ajax

```
220 public function delete_ajax(Request $request, $id){
221     if ($request->ajax() || $request->wantsJson()) {
222         $user = SupplierModel::find($id);
223         if ($user) {
224             $user->delete();
225             return response()->json([
226                 'status' => true,
227                 'message' => 'data berhasil dihapus'
228             ]);
229         } else {
230             return response()->json([
231                 'status' => false,
232                 'message' => 'data tidak ditemukan'
233             ]);
234         }
235     }
236 }
```



- View

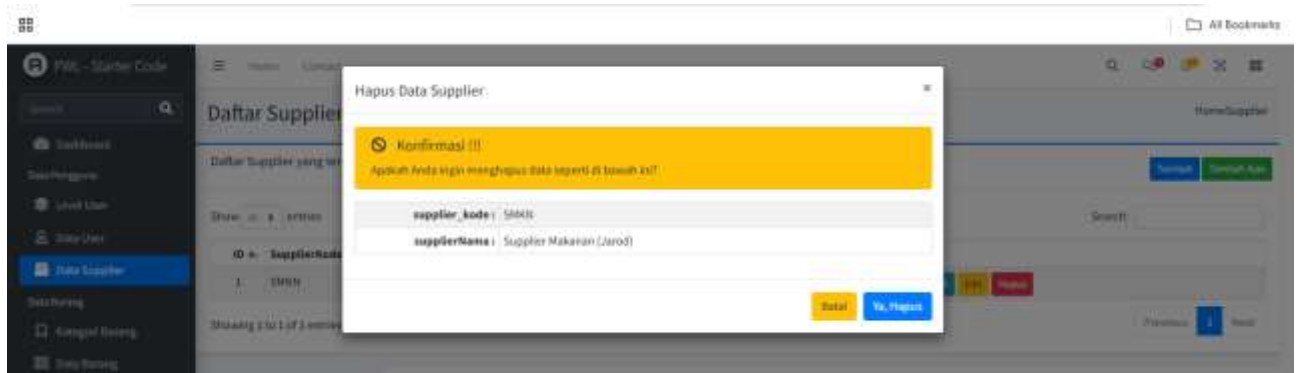
## Create



## Edit



## Delete



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