LAPORAN PRAKTIKUM

JOBSHEET VI: AJAX FORM (ADMINLTE) dan CLIENT VALIDATION

MATA KULIAH PEMROGRAMAN WEB LANJUT

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.

Disusun oleh :

Nama : Vita Eka Saraswati

NIM : 2341760082

No. Absen : 29

JURUSAN TEKNOLOGI INFORMASI

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

POLITEKNIK NEGERI MALANG

2025

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	:	Vita Eka Saraswati
NIM/Kelas	:	2341760082/SIB 2A
No. Absen	:	29
Mata Kuliah	:	Pemrograman Web Lanjut (PWL)

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 from 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	`required`	• Memastikan bahwa field tidak kosong. • Contoh: `required: true`
2	`email`	• Memastikan bahwa input berformat alamat email yang valid. • Contoh: `email: true`
3	`Url`	• Memastikan bahwa input berformat URL yang valid. • Contoh: `url: true`

4	*date*	• Memastikan bahwa input berformat tanggal yang valid (berdasarkan pengaturan regional) • Contoh: `date: true`
5	*dateISO*	• Memastikan bahwa input berformat tanggal yang valid dalam format ISO (YYYY-MM-DD) • Contoh: `dateISO: true`
6	*number*	• Memastikan bahwa input hanya berisi angka (integer atau desimal). • Contoh: `number: true`
7	*digits*	• Memastikan bahwa input hanya berisi angka (tanpa desimal). • Contoh: `digits: true`
8	*creditcard*	• Memastikan bahwa input berformat nomor kartu kredit yang valid. • Contoh: `creditcard: true`
9	*equalTo*	• Memastikan bahwa nilai elemen form sama dengan elemen lain (misalnya, untuk konfirmasi password). • Contoh: `equalTo: "#password"`
10	*maxlength*	• Memastikan bahwa input tidak melebihi jumlah karakter tertentu. • Contoh: `maxlength: 10`
11	*minlength*	• Memastikan bahwa input memiliki minimal jumlah karakter tertentu. • Contoh: `minlength: 5`
12	*rangelength*	• Memastikan bahwa panjang input berada dalam rentang karakter tertentu. • Contoh: `rangelength: [5, 10]`
13	*range*	• Memastikan bahwa nilai input berada dalam rentang tertentu (misalnya, angka 1 sampai 100) • Contoh: `range: [1, 100]`
14	*max*	• Memastikan bahwa nilai input tidak melebihi angka maksimum tertentu. • Contoh: `max: 100`
15	*min*	• Memastikan bahwa nilai input tidak kurang dari angka minimum tertentu. • Contoh: `min: 1`

16	*remote*	• Memvalidasi nilai dengan mengirimkan permintaan ke server untuk memeriksa apakah nilai tersebut valid atau tersedia (misalnya, memeriksa ketersediaan username)
		• Contoh
		`remote: {`
		`  url: "/check-username",`
		`  type: "post"`
		`}`

17	*step*	• Memastikan bahwa nilai input adalah kelipatan dari angka tertentu (berguna untuk validasi angka desimal). • Contoh: `step: 10`
18	*phoneUS*	• Memastikan bahwa input berformat nomor telepon yang valid di AS. • Contoh: `phoneUS: true`

19	*extension*	• Memastikan bahwa file yang diupload memiliki ekstensi tertentu. • Contoh: `extension: "jpg|png|gif"`
20	*accept*	• Memastikan bahwa file yang diupload memiliki jenis MIME tertentu. • Contoh: `accept: "image/*"`
21	*exactlength*	• Memastikan bahwa input hanya berisi karakter yang panjangnya sama persis dengan ketentuan. • Contoh: `exactlength: 10`

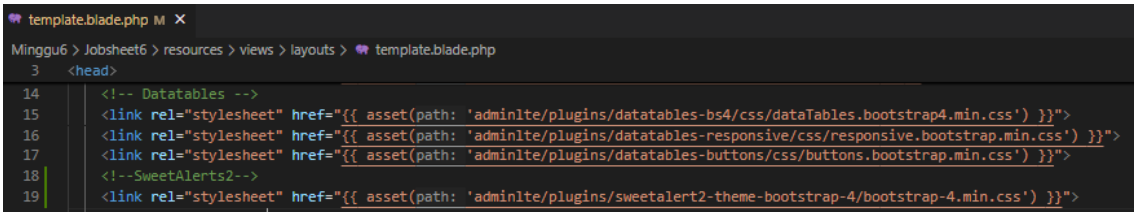
jQuery Validation adalah alat yang sangat berguna untuk memastikan data yang dimasukkan ke dalam form web valid dan sesuai dengan aturan yang ditetapkan sebelum data tersebut dikirim ke server. Ini meningkatkan pengalaman pengguna, mengurangi kesalahan, dan mempermudah pengelolaan validasi form di sisi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

E. Praktikum Jobsheet

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

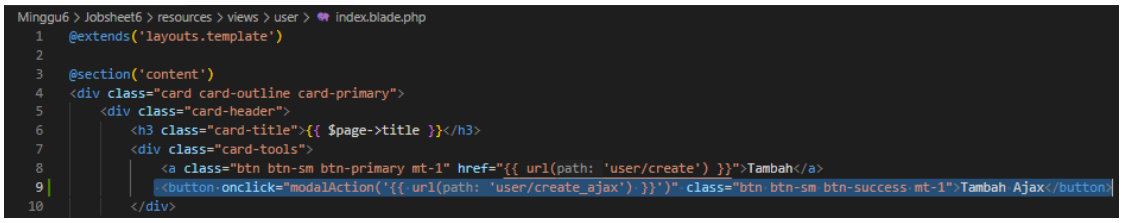
Praktikum 1. Modal Ajax Tambah Data (Data User)

1. Kita buat form tambah data baru dengan menerapkan modal dan proses ajax.
2. 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.



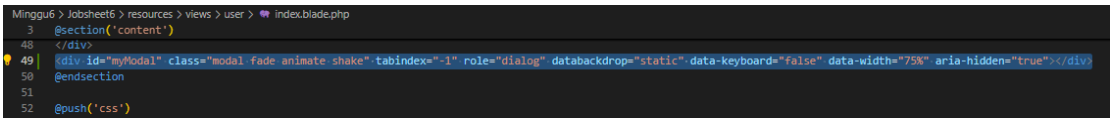
3. Selanjutnya Kita modifikasi view `user/index.blade.php`, kita tambahkan tombol untuk membuat form popup ajax. Kita tambahkan kode berikut, untuk membuat form modal tambah data user dengan ajax

```
<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`

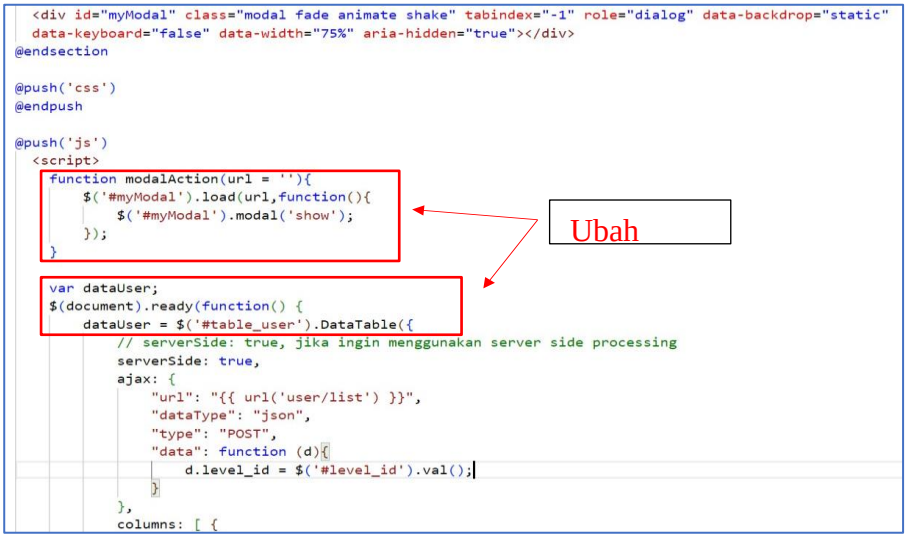
```
<div id="myModal" class="modal fade animate shake" tabindex="-1" role="dialog"
data- backdrop="static" data-keyboard="false" data-width="75%" aria-
hidden="true"></div>
```



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');
    });
}
```

Sehingga tampilan kode program akan seperti berikut



```
Minggu6 > Jobsheet6 > resources > views > user > index.blade.php
55  @push('js')
56  <script>
57      function modalAction(url = ''){
58          $('#myModal').load(url,function(){
59              $('#myModal').modal('show');
60          });
61      }
62
63      var dataUser;
64      $(document).ready(function() {
65          dataUser = $('#table_user').DataTable({
66          // serverSide: true, jika ingin menggunakan server side processing serverSide: true,
67          ajax: {
68              "url": "{{ url(path: 'user/list') }}",
69              "dataType": "json",
70              "type": "POST",
71              "data":function(d){
72                  d.level_id = $('#level_id').val();
73              }
74          },
```

6. Selanjutkan Kita modifikasi `route/web.php` untuk mengakomodir operasi ajax

```
44  //praktikum 3 - no 2
45  Route::group(['prefix' => 'user'], function (): void {
46      Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user
47      Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables
48      Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
49      Route::post('/', [UserController::class, 'store']); // menyimpan data user baru
50
51      //JS6 -PRAK1- route tambah data dengan ajax
52      Route::get('/create_ajax', [UserController::class, 'create_ajax']); // Menampilkan halaman form tambah user Ajax
53      Route::post('/ajax', [UserController::class, 'store_ajax']); // Menyimpan data user baru Ajax
54
55      Route::get('/{id}', [UserController::class, 'show']); // menampilkan detail user
56      Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit user
57      Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
58      Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
59  });
```

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

```
Minggu6 > Jobsheet6 > app > Http > Controllers > UserController.php > PHP > UserController > create
9   class UserController extends Controller
196     //JS6 - PRAK 1 - NO 7
        1 reference | 0 overrides
197     public function create_ajax(): mixed|View
198     {
199         $level = LevelModel::select('level_id', 'level_nama')->get();
200
201         return view(view: 'user.create_ajax')
202             ->with('level', $level);
203     }
204
```

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

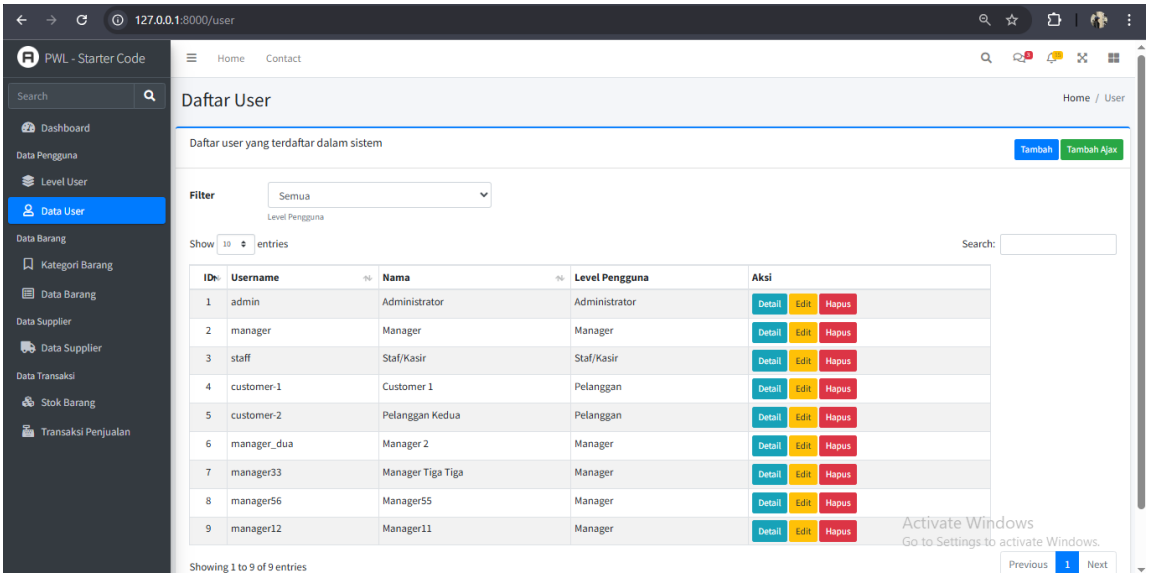
```
Minggu6 > Jobsheet6 > resources > views > user > create_ajax.blade.php
1   <form action="{{ url(path: '/user/ajax') }}" method="POST" id="form-tambah">
2       @csrf
3       <div id="modal-master" class="modal-dialog modal-lg" role="document">
4           <div class="modal-content">
5               <div class="modal-header">
6                   <h5 class="modal-title" id="exampleModalLabel">Tambah Data User</h5>
7                   <button type="button" class="close" data-dismiss="modal" aria-label="Close">
8                       <span aria-hidden="true">&times;</span>
9                   </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 $l)
17                              <option value="{{ $l->level_id }}">{{ $l->level_nama }}</option>
18                          @endforeach
19                      </select>
20                      <small id="error-level_id" class="error-text form-text text-danger"></small>
21                  </div>
22
23                  <div class="form-group">
24                      <label>Username</label>
25                      <input type="text" name="username" id="username" class="form-control" required>
26                      <small id="error-username" class="error-text form-text text-danger"></small>
27                  </div>
28
29                  <div class="form-group">
30                      <label>Nama</label>
31                      <input type="text" name="nama" id="nama" class="form-control" required>
32                      <small id="error-nama" class="error-text form-text text-danger"></small>
33                  </div>
```

https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/resources/views/user/create_ajax.blade.php

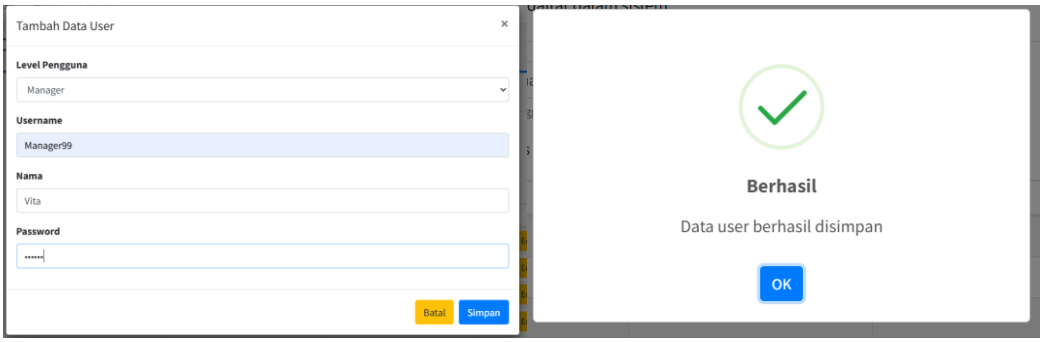
9. Kemudian untuk mengakomodir proses simpan data melalui ajax, kita buat fungsi `store_ajax()` pada `UserController.php`

```
7   use Illuminate\Support\Facades\Validator; // impor untuk function store_ajax
206     public function store_ajax(Request $request): JsonResponse|mixed {
207         // cek apakah request berupa ajax
208         if($request->ajax() || $request->wantsJson()) {
209             $rules = [
210                 'level_id'   => 'required|integer',
211                 'username'   => 'required|string|min:3|unique:m_user,username',
212                 'nama'       => 'required|string|max:100',
213                 'password'   => 'required|min:6'
214             ];
215
216             // use Illuminate\Support\Facades\Validator;
217             $validator = Validator::make($request->all(), $rules);
218
219             if($validator->fails()) {
220                 return response()->json([
221                     'status'   => false, // response status, false: error/gagal, true: berhasil
222                     'message'  => 'Validasi gagal',
223                     'msgField' => $validator->errors() // pesan error validasi
224                 ]);
225             }
226
227             UserModel::create($request->all());
228
229             return response()->json([
230                 'status'  => true,
231                 'message' => 'Data user berhasil disimpan'
232             ]);
233         }
234
235         redirect(to: '/');
236     }
237 }
```

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



Terdapat button berwarna hijau dengan teks “tambah ajax”. Saat diklik akan muncul form pop up tambah data dan pesan sukses simpan data seperti di bawah ini:



9	manager12	Manager11	Manager	Detail Edit Hapus
10	Manager99	Vita	Manager	Detail Edit Hapus

Showing 1 to 10 of 10 entries

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

```php
// Ambil data user dalam bentuk json untuk datatables
public function list(Request $request): JsonResponse
{
    $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): string { // menambahkan kolom aksi
            // $btn = '<a href="'.url('/user/' . $user->user_id).'" class="btn btn-info btn-sm">Detail</a> ';
            // $btn .= '<a href="'.url('/user/' . $user->user_id . '/edit').'" 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(path: '/user/' . $user->user_id . '/show_ajax').'\')" class="btn btn-info btn-sm">Detail</button> ';
            $btn .= '<button onclick="modalAction(\''.url(path: '/user/' . $user->user_id . '/edit_ajax').'\')" class="btn btn-warning btn-sm">Edit</button> ';
            $btn .= '<button onclick="modalAction(\''.url(path: '/user/' . $user->user_id . '/delete_ajax').'\')" class="btn btn-danger btn-sm">Hapus</button> ';
            return $btn;
        })
        ->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html
        ->make(true);
}
```

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

```php
//praktikum 3 - no 2
Route::group(['prefix' => 'user'], function (): void {
    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

    //JS6 -PRAK1- route tambah data dengan ajax
    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
    //JS6 -PRAK1- route edit data dengan ajax
    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}', [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
});
```

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

```php
    }

    //Menampilkan halaman edit user dengan ajax
    1 reference | 0 overrides
    public function edit_ajax(string $id): Factory|View
    {
        $user = UserModel::find($id);
        $level = LevelModel::select('level_id', 'level_nama')->get();

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

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

Minggu6 > Jobsheet6 > resources > views > user > edit_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(path: '/user') }}" class="btn btn-warning">Kembali</a>
        </div>
    </div>
</div>
@else
    <form action="{{ url(path: '/user/' . $user->user_id . '/update_ajax') }}" method="POST" id="form-edit">
        @csrf
        @method('PUT')
```

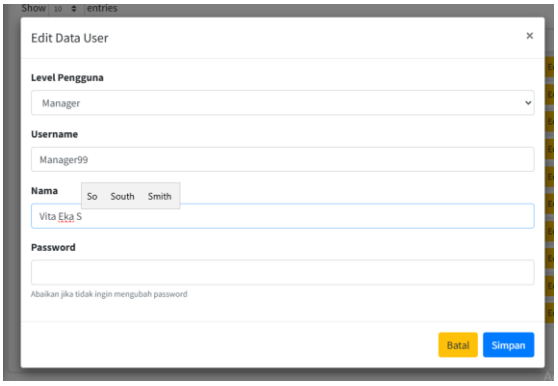
Selengkapnya :

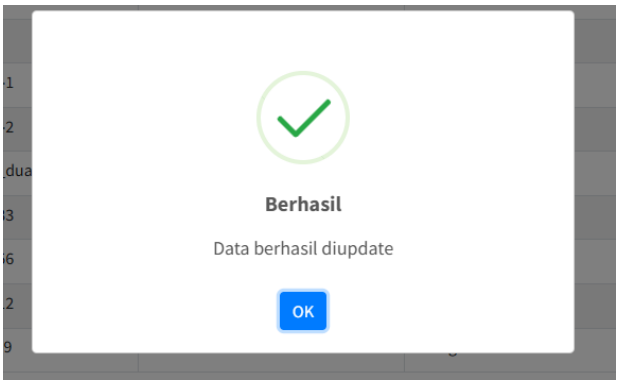
https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/resources/views/user/edit_ajax.blade.php

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

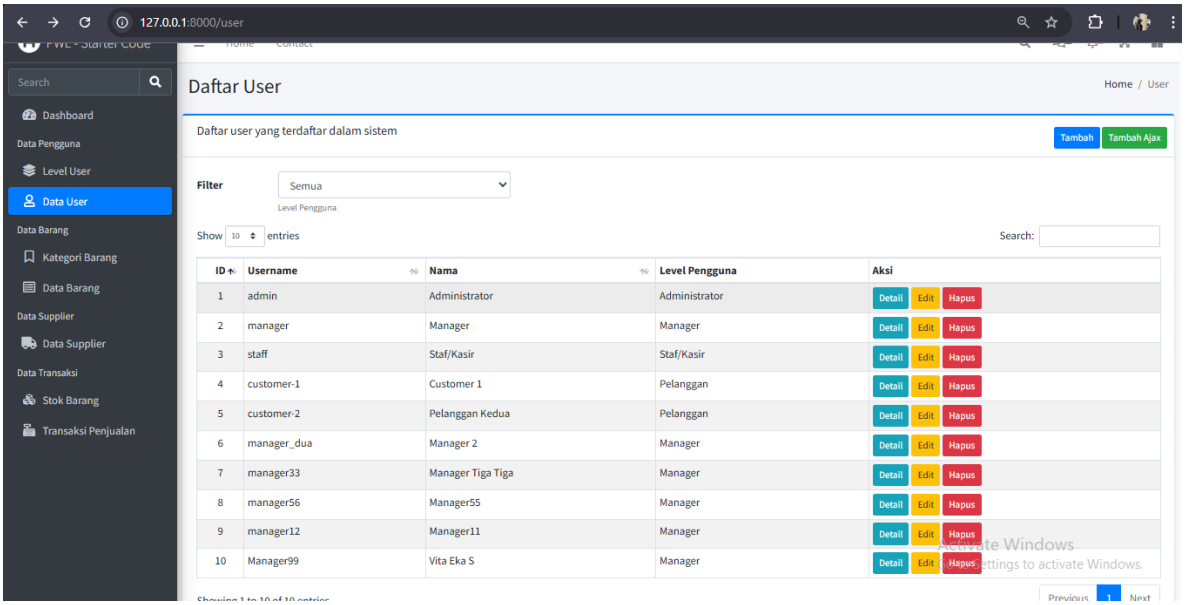
```
// Mengupdate data user dengan ajax
1 reference | 0 overrides
public function update_ajax(Request $request, $id): JsonResponse|mixed|Redirector|RedirectRes…
    // Cek apakah request dari AJAX atau JSON
    if ($request->ajax() || $request->wantsJson()) {
        $rules = [
            'level_id' => 'required|integer',
            'username' => 'required|max:20|unique:m_user,username,' . $id . ',user_id',
            'nama' => 'required|max:100',
            'password' => 'nullable|min:6|max:20'
        ];

        $validator = Validator::make($request->all(), $rules);

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

    $check = UserModel::find($id);
    if ($check) {
        if (!$request->filled('password')) { // Jika password tidak diisi, 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'
        ]);
    }
    }
    // Redirect jika request tidak menggunakan AJAX
    return redirect(to: '/');
    }
}
```

7. Sekarang kita coba bagian edit user, amati proses nya. Jangan lupa laporkan dan *commit* ke *repository git* kalian !





Setelah user mengklik button simpan maka akan muncul pop up alert pesan sukses diupdate



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

```php
Route::group(['prefix' => 'user'], function (): void {
    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

    //JS6 -PRAK1- route tambah data dengan ajax
    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
    //JS6 -PRAK2- route edit data dengan ajax
    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}', [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

    //JS6 -PRAK3- route hapus data dengan ajax
    Route::get('/{id}/delete_ajax', [UserController::class, 'confirm_ajax']); //Untuk tampilkan 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
});
```

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

```php
    //Konfirmasi hapus data user
    1 reference | 0 overrides
    public function confirm_ajax(string $id): Factory|View
    {
        $user = UserModel::find($id);

        return view(view: 'user.delete_ajax', data: ['user' => $user]);
    }
}
```

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

```
Minggu6 > Jobsheet6 > resources > views > 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(path: '/user') }}" class="btn btn-warning">Kembali</a>
            </div>
        </div>
    </div>
@else
    <form action="{{ url(path: '/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>
```

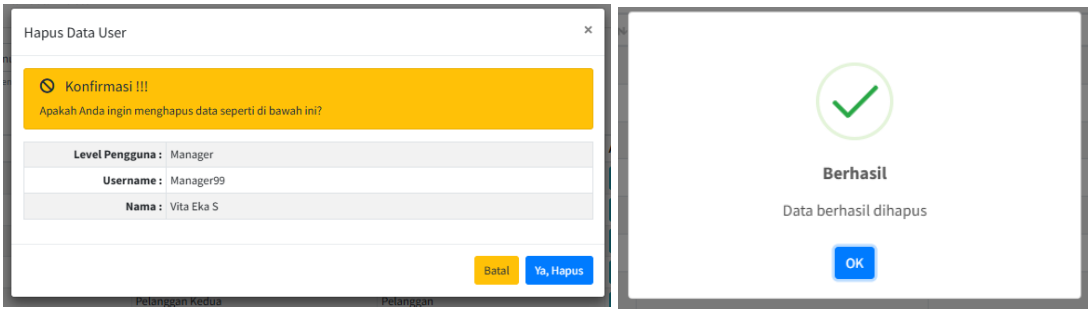
Selengkapnya :

https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/resources/views/user/confirm_ajax.blade.php

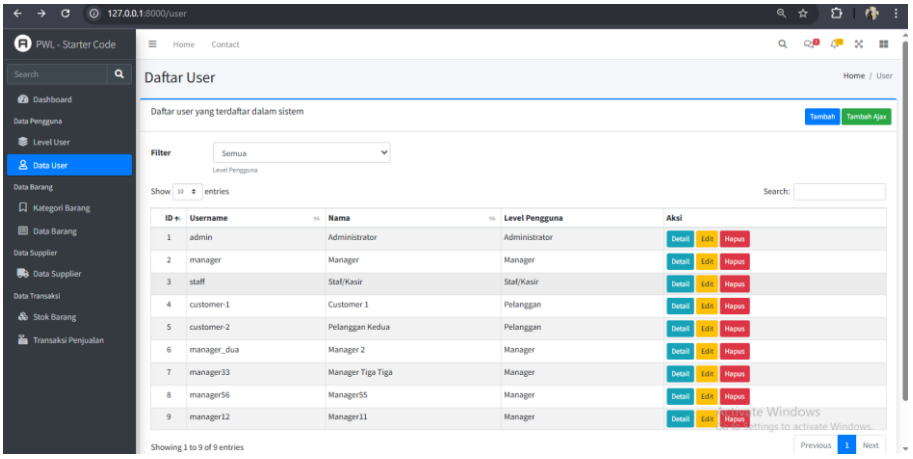
5. Kemudian kita buat fungsi `delete_ajax()` pada `UserController.php` untuk mengakomodir *request* hapus data user

```php
    public function delete_ajax(Request $request, $id): JsonResponse|mixed|Redirector|RedirectRes…
    {
    // cek apakah request dari ajax
    if ($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(to: '/');
    }
}
```

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.!!!



Saat user mengklik button “Hapus data” maka akan muncul konfirmasi pop up dengan memuat detail informasi data yang akan di hapus seperti pada gambar di atas.



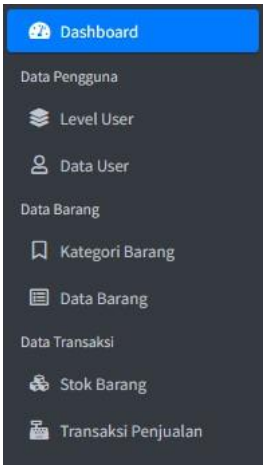
Tampilan setelah data di hapus

F. Tugas Jobsheet

Implementasikan koding untuk Ajax Form dan Client Validation dengan jQuery Validation pada 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.



Jawab :

1. Tabel m_level

a. Tambahkan route ajax pada web.php

```php
//tugas 1 m_level
Route::group(['prefix' => 'level'], function (): void {
    Route::get('/', [LevelController::class, 'index']); // menampilkan halaman awal level
    Route::post("/list", [LevelController::class, 'list']); // menampilkan data level dalam bentuk json untuk datatables
    Route::get('/create', [LevelController::class, 'create']); // menampilkan halaman form tambah level
    Route::post('/', [LevelController::class, 'store']); // menyimpan data level baru

    //JS6 -TUGAS1- route tambah data level dengan ajax
    Route::get('/create_ajax', [LevelController::class, 'create_ajax']); // Menampilkan halaman form tambah level Ajax
    Route::post('/ajax', [LevelController::class, 'store_ajax']); // Menyimpan data level baru Ajax
    //JS6 -TUGAS1- route edit data level dengan ajax
    Route::get('/{id}/edit_ajax', [LevelController::class, 'edit_ajax']); // Menampilkan halaman form edit level Ajax
    Route::put('/{id}/update_ajax', [LevelController::class, 'update_ajax']); // Menyimpan perubahan data level Ajax

    Route::get('/{id}', [LevelController::class, 'show']); // menampilkan detail level
    Route::get('/{id}/edit', [LevelController::class, 'edit']); // menampilkan halaman form edit level
    Route::put('/{id}', [LevelController::class, 'update']); // menyimpan perubahan data level

    //JS6 -TUGAS1- route hapus data level dengan ajax
    Route::get('/{id}/delete_ajax', [LevelController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete level Ajax
    Route::delete('/{id}/delete_ajax', [LevelController::class, 'delete_ajax']); // Untuk hapus data level Ajax
    Route::delete('/{id}', [LevelController::class, 'destroy']); // menghapus data level
});
```

b. Modifikasi function list pada file LevelController.php

```php
public function list(Request $request): JsonResponse
{
    $levels = LevelModel::select('level_id', 'level_kode', 'level_nama');

    if ($request->level_id) {
        $levels->where('level_id', $request->level_id);
    }

    return DataTables::of($levels)
        ->addIndexColumn() // menambahkan kolom index / no urut (default nama kolom: DT_RowIndex)
        ->addColumn('aksi', function ($level): string { // menambahkan kolom aksi
            // $btn = '<a href="' . url('/level/' . $level->level_id) . '" class="btn btn-info btn-sm">Detail</a> ';
            // $btn .= '<a href="' . url('/level/' . $level->level_id . '/edit') . '" class="btn btn-warning btn-sm">Edit</a> ';
            // $btn .= '<form class="d-inline-block" method="POST" action="' . url('/level/' . $level->level_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(path: '/level/' . $level->level_id . '/show_ajax') . '\')" class="btn btn-info btn-sm">Detail</button> ';
            $btn .= '<button onclick="modalAction(\'' . url(path: '/level/' . $level->level_id . '/edit_ajax') . '\')" class="btn btn-warning btn-sm">Edit</button> ';
            $btn .= '<button onclick="modalAction(\'' . url(path: '/level/' . $level->level_id . '/delete_ajax') . '\')" class="btn btn-danger btn-sm">Hapus</button> ';
            return $btn;
        })
        ->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html
        ->make(true);
}
```

c. Modifikasi file views/level/index.blade.php

```
Minggu6 > Jobsheet6 > resources > views > level > index.blade.php
@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 href="{{ url(path: 'level/create') }}" class="btn btn-sm btn-primary mt-1">Tambah</a>
                <button onclick="modalAction('{{ url(path: 'level/create_ajax') }}')" class="btn btn-sm btn-success mt-1">Tambah Ajax</button>
            </div>
```

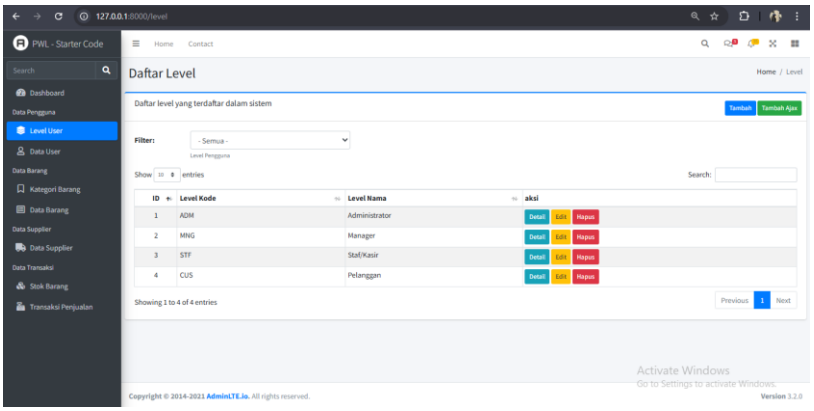
```
            </table>
        </div>
    </div>
</div>
<div id="myModal" class="modal fade animate shake" tabindex="-1" role="dialog" data- backdrop="static" data-keyboard="false" data-width="75%" aria-hidden="true"></div>
@endsection

@push('css')
@endpush
```

```
@push('css')
@endpush

@push('js')
    <script>
        function modalAction(url = ''){
            $('#myModal').load(url,function(){
                $('#myModal').modal('show')
            });
        }
        $(document).ready(function(){
            var dataLevel = $('#table_level').DataTable({
```

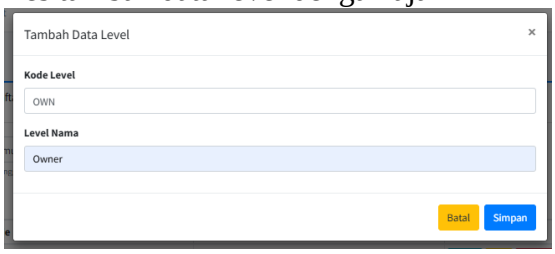
d. Modifikasi file controller LevelController.php dengan menambahkan function create_ajax, store_ajax, edit_ajax, update_ajax, confirm_ajax, delete_ajax
https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/app/Http/Controllers/LevelController.php

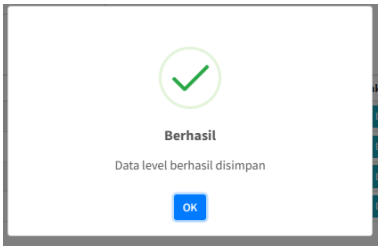
e. Buat file view create_ajax, edit_ajax, dan confirm_ajax blade (pada folder resources/views/level
https://github.com/vitasaraswati/PWL25/tree/master/Minggu6/Jobsheet6/resources/views/level

f. Tampilan pada halaman web browser



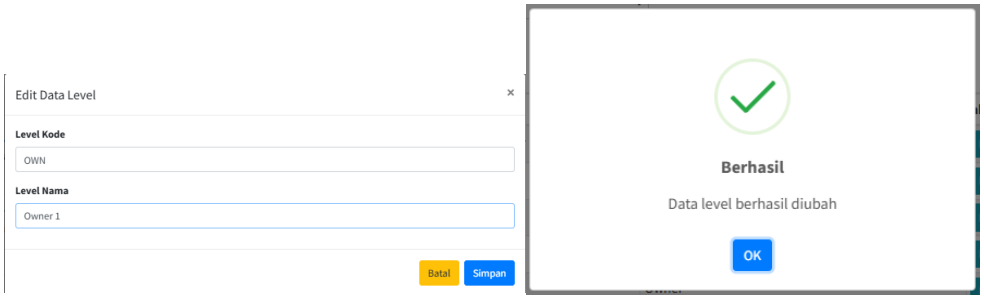
- Tes tambah data level dengan ajax





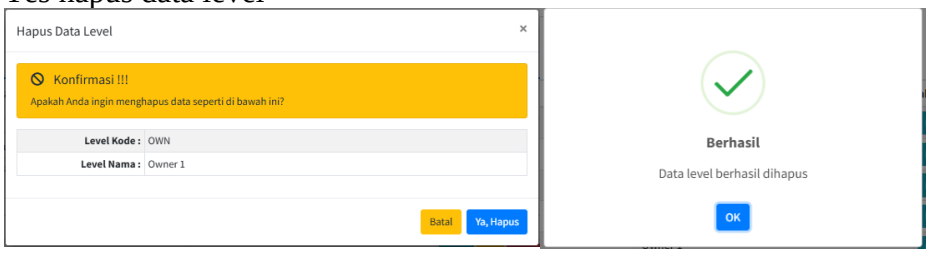
ID	Level Kode	Level Nama	aksi
1	ADM	Administrator	Detail Edit Hapus
2	MNG	Manager	Detail Edit Hapus
3	STF	Staf/Kasir	Detail Edit Hapus
4	CUS	Pelanggan	Detail Edit Hapus
5	OWN	Owner	Detail Edit Hapus

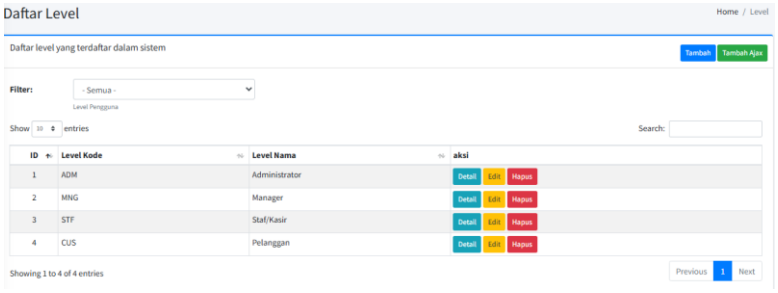
- Tes edit data level



ID	Level Kode	Level Nama	aksi
1	ADM	Administrator	Detail Edit Hapus
2	MNG	Manager	Detail Edit Hapus
3	STF	Staf/Kasir	Detail Edit Hapus
4	CUS	Pelanggan	Detail Edit Hapus
5	OWN	Owner 1	Detail Edit Hapus

- Tes hapus data level





2. Tabel m_kategori

a. Tambahkan route ajax pada web.php

```php
//tugas 2 m_kategori
Route::group(['prefix' => 'kategori'], function (): void {
    Route::get('/', [KategoriController::class, 'index']); // menampilkan halaman awal kategori
    Route::post("/list", [KategoriController::class, 'list']); // menampilkan data kategori dalam bentuk json untuk datatables
    Route::get('/create', [KategoriController::class, 'create']); // menampilkan halaman form tambah kategori
    Route::post('/', [KategoriController::class, 'store']); // menyimpan data kategori baru

    //JS6 -TUGAS2- route tambah data kategori dengan ajax
    Route::get('/create_ajax', [KategoriController::class, 'create_ajax']); // Menampilkan halaman form tambah kategori Ajax
    Route::post('/ajax', [KategoriController::class, 'store_ajax']); // Menyimpan data kategori baru Ajax
    //JS6 -TUGAS2- route edit data kategori dengan ajax
    Route::get('/{id}/edit_ajax', [KategoriController::class, 'edit_ajax']); // Menampilkan halaman form edit kategori Ajax
    Route::put('/{id}/update_ajax', [KategoriController::class, 'update_ajax']); // Menyimpan perubahan data kategori Ajax

    Route::get('/{id}', [KategoriController::class, 'show']); // menampilkan detail kategori
    Route::get('/{id}/edit', [KategoriController::class, 'edit']); // menampilkan halaman form edit kategori
    Route::put('/{id}', [KategoriController::class, 'update']); // menyimpan perubahan data kategori

    //JS6 -TUGAS2- route hapus data kategori dengan ajax
    Route::get('/{id}/delete_ajax', [KategoriController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete kategori Ajax
    Route::delete('/{id}/delete_ajax', [KategoriController::class, 'delete_ajax']); // Untuk hapus data kategori Ajax

    Route::delete('/{id}', [KategoriController::class, 'destroy']); // menghapus data kategori
});
```

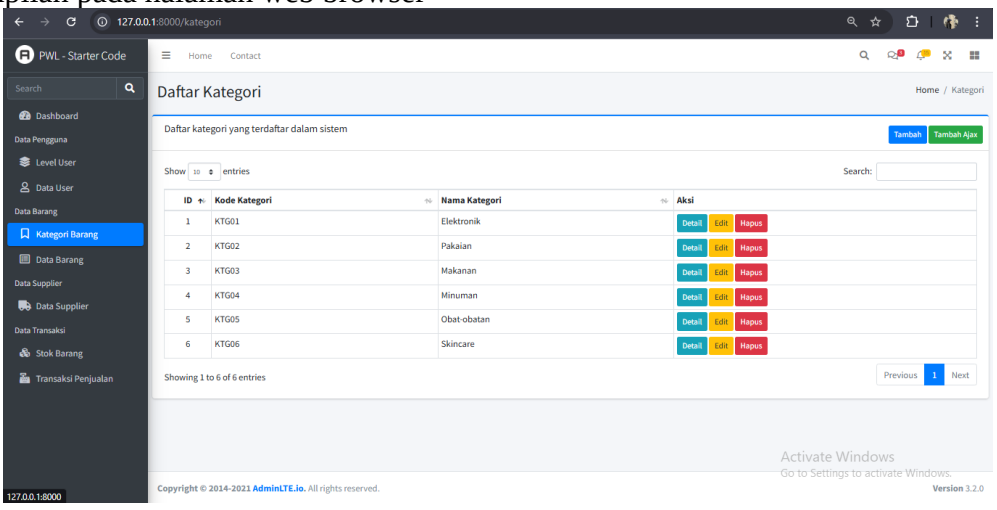
b. Modifikasi function list pada file KategoriController.php

c. Modifikasi file views/kategori/index.blade.php

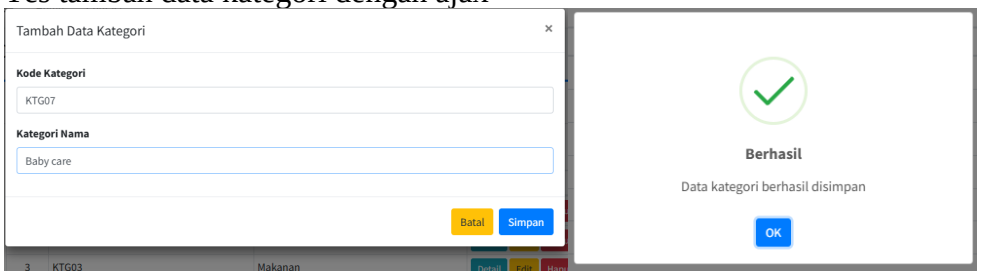
d. Modifikasi file controller KategoriController.php dengan menambahkan function create_ajax, store_ajax, edit_ajax, update_ajax, confirm_ajax, delete_ajax
https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/app/Http/Controllers/KategoriController.php

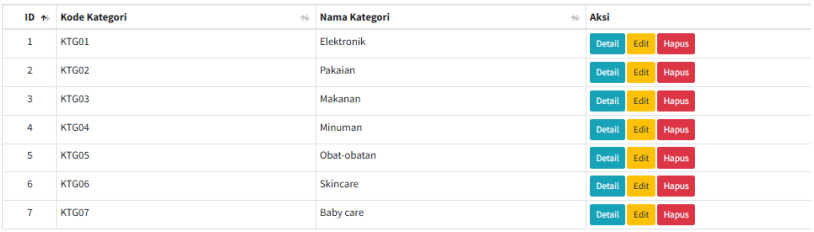
e. Buat file view create_ajax, edit_ajax, dan confirm_ajax blade (pada folder resources/views/kategori
https://github.com/vitasaraswati/PWL25/tree/master/Minggu6/Jobsheet6/resources/views/kategori

f. Tampilan pada halaman web browser



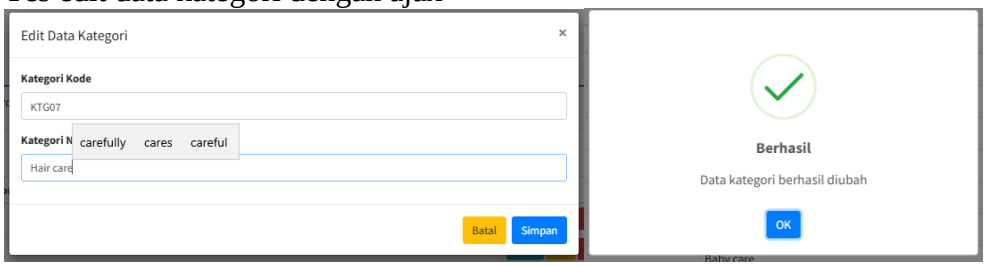
- Tes tambah data kategori dengan ajax

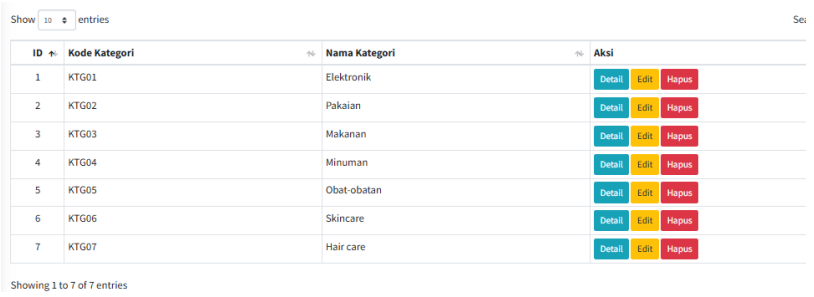




ID	Kode Kategori	Nama Kategori	Aksi
1	KTG01	Elektronik	Detail Edit Hapus
2	KTG02	Pakaian	Detail Edit Hapus
3	KTG03	Makanan	Detail Edit Hapus
4	KTG04	Minuman	Detail Edit Hapus
5	KTG05	Obat-obatan	Detail Edit Hapus
6	KTG06	Skincare	Detail Edit Hapus
7	KTG07	Baby care	Detail Edit Hapus

- Tes edit data kategori dengan ajax



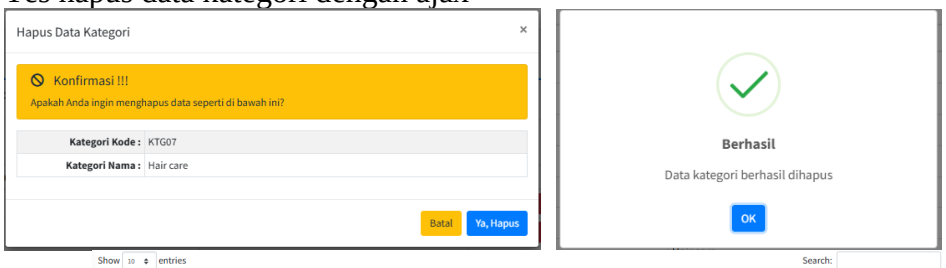


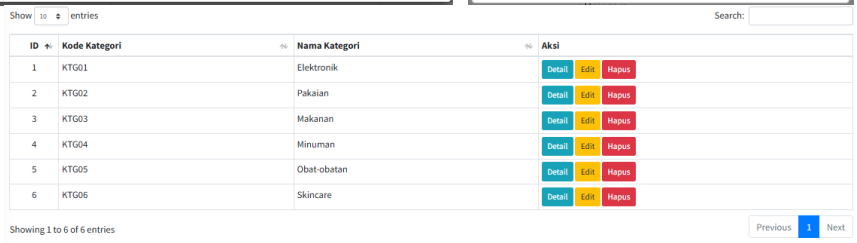
Show 10 entries

ID	Kode Kategori	Nama Kategori	Aksi
1	KTG01	Elektronik	Detail Edit Hapus
2	KTG02	Pakaian	Detail Edit Hapus
3	KTG03	Makanan	Detail Edit Hapus
4	KTG04	Minuman	Detail Edit Hapus
5	KTG05	Obat-obatan	Detail Edit Hapus
6	KTG06	Skincare	Detail Edit Hapus
7	KTG07	Hair care	Detail Edit Hapus

Showing 1 to 7 of 7 entries

- Tes hapus data kategori dengan ajax





Show 10 entries Search:

ID	Kode Kategori	Nama Kategori	Aksi
1	KTG01	Elektronik	Detail Edit Hapus
2	KTG02	Pakaian	Detail Edit Hapus
3	KTG03	Makanan	Detail Edit Hapus
4	KTG04	Minuman	Detail Edit Hapus
5	KTG05	Obat-obatan	Detail Edit Hapus
6	KTG06	Skincare	Detail Edit Hapus

Showing 1 to 6 of 6 entries Previous 1 Next

3. Tabel m_supplier

a. Tambahkan route ajax pada web.php

```php
//tugas 3 m_supplier
Route::group(['prefix' => 'supplier'], function (): void {
    Route::get('/', [SupplierController::class, 'index']); // menampilkan halaman awal supplier
    Route::post("/list", [SupplierController::class, 'list']); // menampilkan data supplier dalam bentuk json untuk datatables
    Route::get('/create', [SupplierController::class, 'create']); // menampilkan halaman form tambah supplier
    Route::post('/', [SupplierController::class, 'store']); // menyimpan data supplier baru

    //JS6 -TUGAS3- route tambah data supplier dengan ajax
    Route::get('/create_ajax', [SupplierController::class, 'create_ajax']); // Menampilkan halaman form tambah supplier Ajax
    Route::post('/ajax', [SupplierController::class, 'store_ajax']); // Menyimpan data supplier baru Ajax
    //JS6 -TUGAS3- route edit data supplier dengan ajax
    Route::get('/{id}/edit_ajax', [SupplierController::class, 'edit_ajax']); // Menampilkan halaman form edit supplier Ajax
    Route::put('/{id}/update_ajax', [SupplierController::class, 'update_ajax']); // Menyimpan perubahan data supplier Ajax

    Route::get('/{id}', [SupplierController::class, 'show']); // menampilkan detail supplier
    Route::get('/{id}/edit', [SupplierController::class, 'edit']); // menampilkan halaman form edit supplier
    Route::put('/{id}', [SupplierController::class, 'update']); // menyimpan perubahan data supplier

    //JS6 -TUGAS3- route hapus data supplier dengan ajax
    Route::get('/{id}/delete_ajax', [SupplierController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete supplier Ajax
    Route::delete('/{id}/delete_ajax', [SupplierController::class, 'delete_ajax']); // Untuk hapus data supplier Ajax

    Route::delete('/{id}', [SupplierController::class, 'destroy']); // menghapus data supplier
});
```

b. Modifikasi function list pada file SupplierController.php

```php
// Ambil data supplier dalam bentuk json untuk datatables
public function list(Request $request): JsonResponse
{
    $supplier = SupplierModel::select('supplier_id', 'supplier_kode', 'supplier_nama', 'supplier_alamat');

    if ($request->supplier_id) {
        $supplier->where('supplier_id', $request->supplier_id);
    }

    return DataTables::of($supplier)
        ->addIndexColumn() // menambahkan kolom index
        ->addColumn('aksi', function ($supplier): string {
            // $btn = '<a href="' . url('/supplier/' . $supplier->supplier_id . '/edit') . '" class="btn btn-info btn-sm">Detail</a> ';
            // $btn .= '<a href="' . url('/supplier/' . $supplier->supplier_id . '/edit') . '" class="btn btn-warning btn-sm">Edit</a> ';
            // $btn .= '<form class="d-inline-block" method="POST" action="' . url('/supplier/' . $supplier->supplier_id) . '">' . csrf_field() . method_field('DELETE') . '<butt
            $btn = '<button onclick="modalAction(\'' . url(path: '/supplier/' . $supplier->supplier_id . '/show_ajax') . '\')" class="btn btn-info btn-sm">Detail</button> ';
            $btn .= '<button onclick="modalAction(\'' . url(path: '/supplier/' . $supplier->supplier_id . '/edit_ajax') . '\')" class="btn btn-warning btn-sm">Edit</button> ';
            $btn .= '<button onclick="modalAction(\'' . url(path: '/supplier/' . $supplier->supplier_id . '/delete_ajax') . '\')" class="btn btn-danger btn-sm">Hapus</button> ';
            return $btn;
        })
        ->rawColumns(['aksi'])
        ->make(true);
}
```

c. Modifikasi file views/supplier/index.blade.php

```
Minggu6 > Jobsheet6 > resources > views > supplier > index.blade.php
@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(path: 'supplier/create') }}">Tambah</a>
                <button onclick="modalAction('{{ url(path: 'supplier/create_ajax') }}')" class="btn btn-sm btn-success mt-1">Tambah Ajax</button>
            </div>
        </div>
```

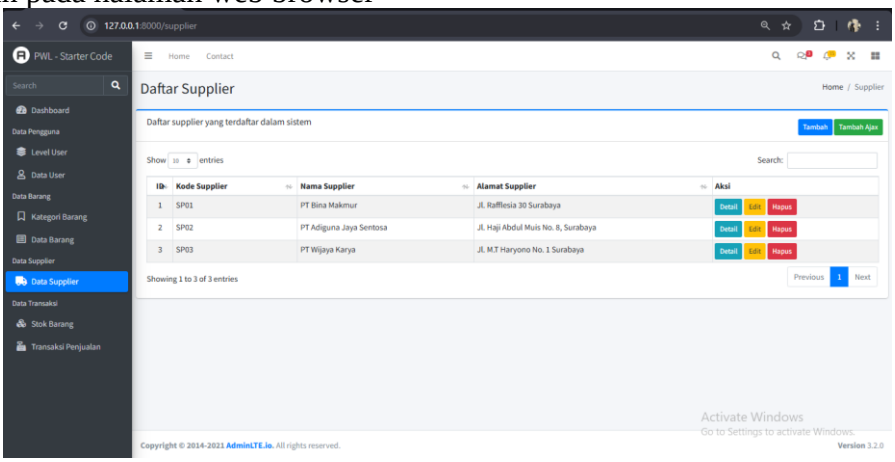
```
        </div>
    </div>
<div id="myModal" class="modal fade animate shake" tabindex="-1" role="dialog" data-backdrop="static" data-keyboard="false" data-width="75%" aria-hidden="true"></div>
@endsection

@push('css')
@endpush

@push('js')
```

```
@push('js')
    <script>
        function modalAction(url = ''){
            $('#myModal').load(url,function(){
            $('#myModal').modal('show')
        });
    }
        $(document).ready(function() {
            var dataKategori = $('#table_supplier').DataTable({
```

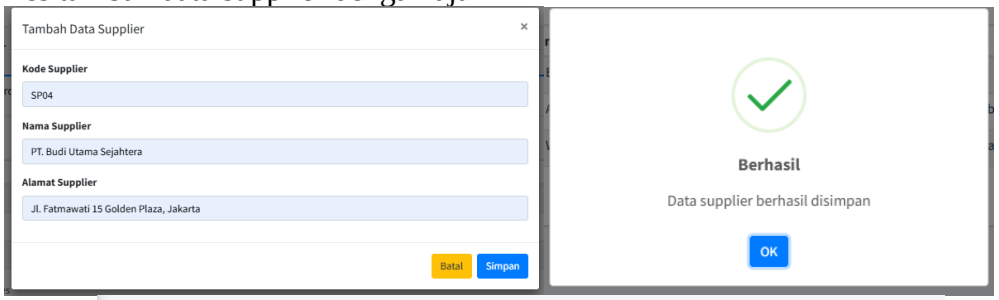
d. Modifikasi file controller SupplierController.php dengan menambahkan function create_ajax, store_ajax, edit_ajax, update_ajax, confirm_ajax, delete_ajax
https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/app/Http/Controllers/SupplierController.php

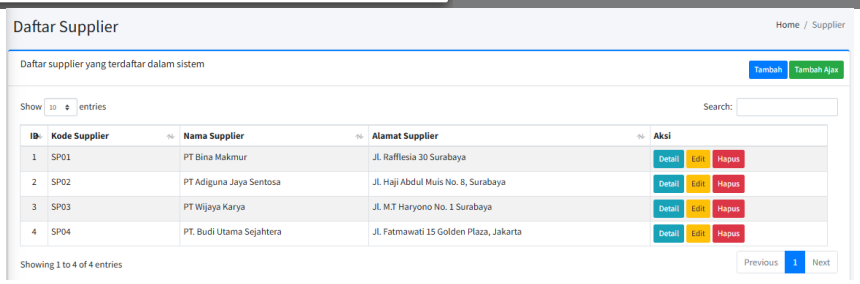
e. Buat file view create_ajax, edit_ajax, dan confirm_ajax blade (pada folder resources/views/supplier
https://github.com/vitasaraswati/PWL25/tree/master/Minggu6/Jobsheet6/resources/views/supplier

f. Tampilan pada halaman web browser

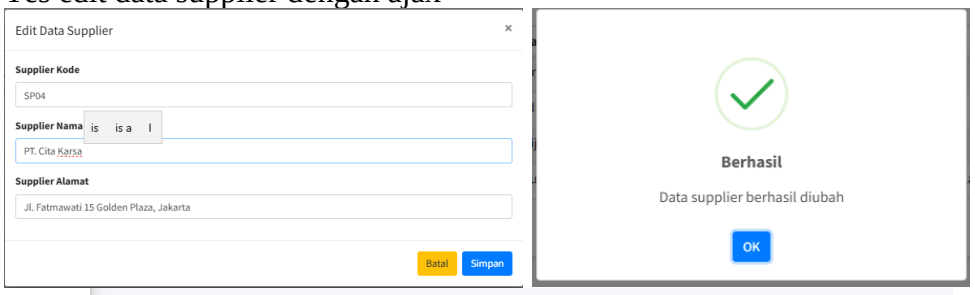


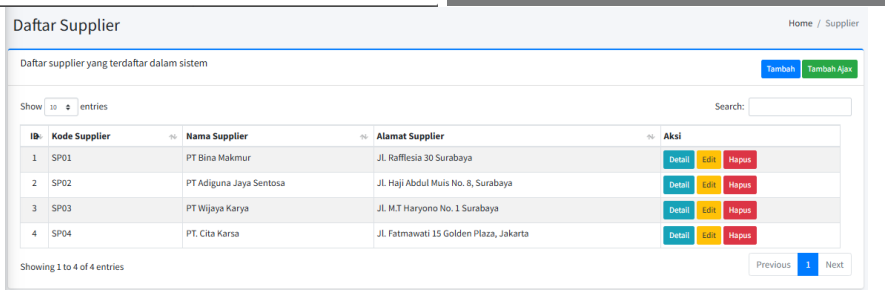
- Tes tambah data supplier dengan ajax



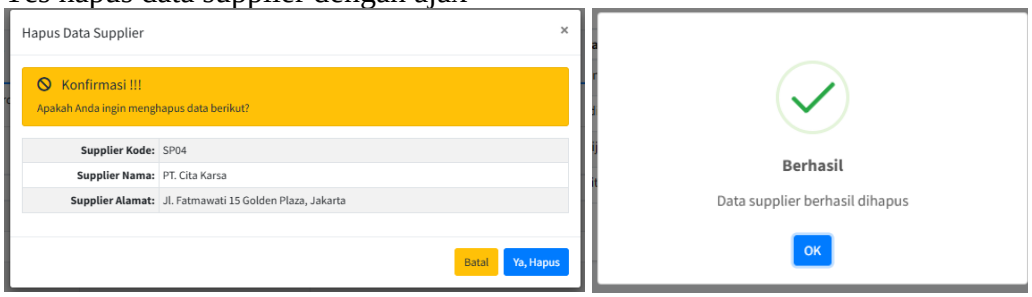


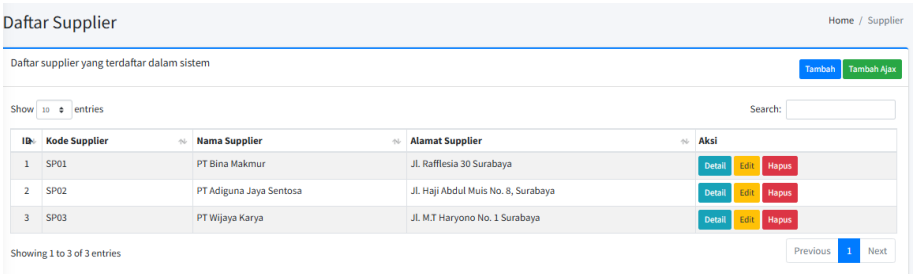
- Tes edit data supplier dengan ajax





- Tes hapus data supplier dengan ajax





4. Tabel m_barang

a. Tambahkan route ajax pada web.php

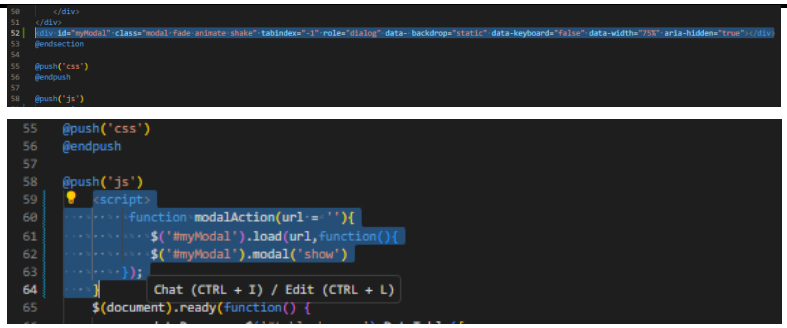
```php
//tugas 4 m_barang
Route::group(['prefix' => 'barang'], function (): void {
    Route::get('/', [BarangController::class, 'index']); // menampilkan halaman awal barang
    Route::post('/list', [BarangController::class, 'list']); // menampilkan data barang dalam bentuk json untuk datatables
    Route::get('/create', [BarangController::class, 'create']); // menampilkan halaman form tambah barang
    Route::post('/', [BarangController::class, 'store']); // menyimpan data barang baru

    //JS6 -TUGAS4- route tambah data barang dengan ajax
    Route::get('/create_ajax', [BarangController::class, 'create_ajax']); // Menampilkan halaman form tambah barang Ajax
    Route::post('/ajax', [BarangController::class, 'store_ajax']); // Menyimpan data barang baru Ajax
    //JS6 -TUGAS4- route edit data barang dengan ajax
    Route::get('/{id}/edit_ajax', [BarangController::class, 'edit_ajax']); // Menampilkan halaman form edit barang Ajax
    Route::put('/{id}/update_ajax', [BarangController::class, 'update_ajax']); // Menyimpan perubahan data barang Ajax

    Route::get('/{id}', [BarangController::class, 'show']); // menampilkan detail barang
    Route::get('/{id}/edit', [BarangController::class, 'edit']); // menampilkan halaman form edit barang
    Route::put('/{id}', [BarangController::class, 'update']); // menyimpan perubahan data barang

    //JS6 -TUGAS4- route hapus data barang dengan ajax
    Route::get('/{id}/delete_ajax', [BarangController::class, 'confirm_ajax']); // Untuk tampilkan form confirm delete barang Ajax
    Route::delete('/{id}/delete_ajax', [BarangController::class, 'delete_ajax']); // Untuk hapus data barang Ajax

    Route::delete('/{id}', [BarangController::class, 'destroy']); // menghapus data barang
});
```

b. Modifikasi function list pada file BarangController.php

```php
public function list(Request $request): JsonResponse
{
    $barangs = BarangModel::select('barang_id', 'barang_kode', 'barang_nama', 'harga_beli', 'harga_jual', 'kategori_id')
        ->with('kategori');

    // Filter data barang berdasarkan kategori_id
    if ($request->kategori_id) {
        $barangs->where('kategori_id', $request->kategori_id);
    }

    return DataTables::of($barangs)
        ->addIndexColumn() // menambahkan kolom index / no urut (default nama kolom: DT_RowIndex)
        ->addColumn('aksi', function ($barang): string { // menambahkan kolom aksi
            // $btn = '<a href="' . url('/barang/' . $barang->barang_id . '/edit') . '" class="btn btn-warning btn-sm">Edit</a> ';
            // $btn .= '<form class="d-inline-block" method="POST" action="' .url('/barang/' . $barang->barang_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(path: '/barang/' . $barang->barang_id . '/show_ajax') . '\')" class="btn btn-info btn-sm">Detail</button> ';
            $btn .= '<button onclick="modalAction(\'' . url(path: '/barang/' . $barang->barang_id . '/edit_ajax') . '\')" class="btn btn-warning btn-sm">Edit</button> ';
            $btn .= '<button onclick="modalAction(\'' . url(path: '/barang/' . $barang->barang_id . '/delete_ajax') . '\')" class="btn btn-danger btn-sm">Hapus</button> ';
            return $btn;
        })
        ->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html
        ->make(true);
}
```

c. Modifikasi file views/barang/index.blade.php

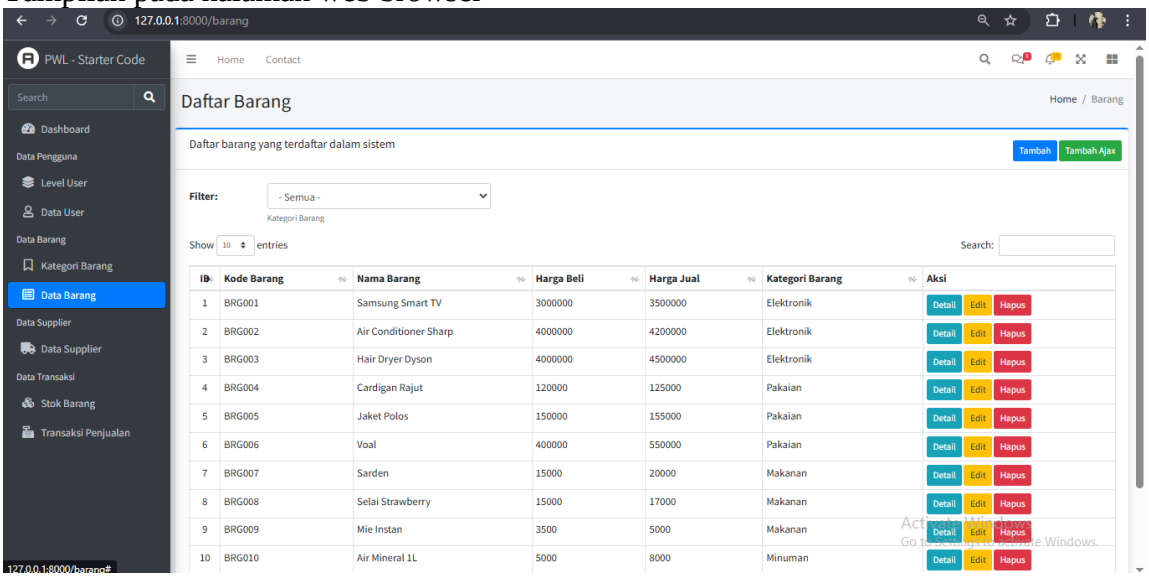
```
Minggu6 > Jobsheet6 > resources > views > barang > index.blade.php
@extends('layouts.template')

@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(path: 'barang/create') }}">Tambah</a>
            <button onclick="modalAction('{{ url(path: 'barang/create_ajax') }}')" class="btn btn-sm btn-success mt-1">Tambah Ajax</button>
        </div>
    </div>
```



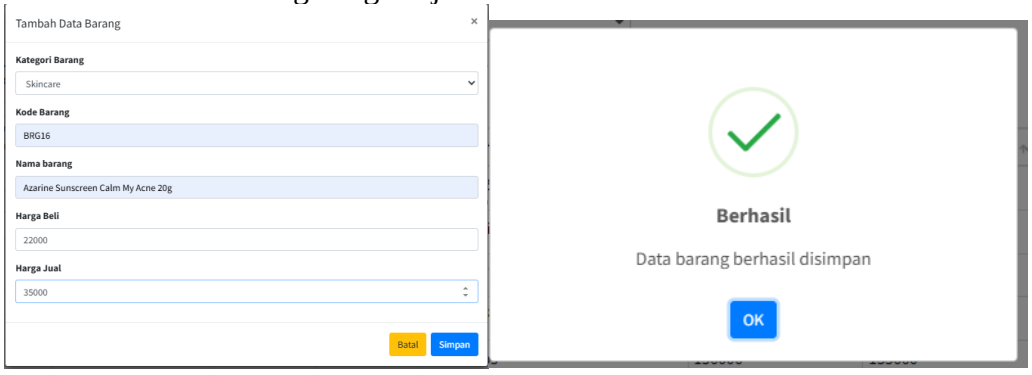
d. Modifikasi file controller BarangController.php dengan menambahkan function create_ajax, store_ajax, edit_ajax, update_ajax, confirm_ajax, delete_ajax
https://github.com/vitasaraswati/PWL25/blob/master/Minggu6/Jobsheet6/app/Http/Controllers/BarangController.php

e. Buat file view create_ajax, edit_ajax, dan confirm_ajax blade (pada folder resources/views/barang
https://github.com/vitasaraswati/PWL25/tree/master/Minggu6/Jobsheet6/resources/views/barang

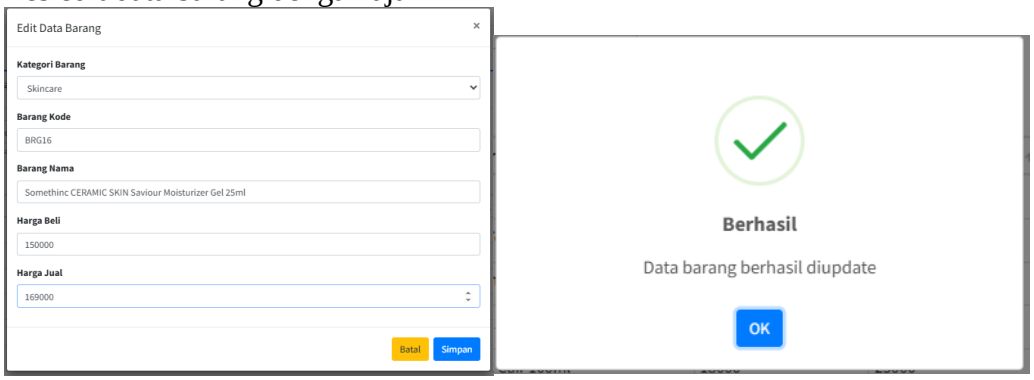
f. Tampilan pada halaman web browser



- Tes tambah data barang dengan ajax



- Tes edit data barang dengan ajax



- Tes hapus data barang dengan ajax