



Nama : Arimbi Putri Hapsari
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?

- Response Instan:** AJAX memungkinkan Kita untuk mengirim data dan menerima respons dari server tanpa perlu memuat ulang halaman.
- Pengalaman Pengguna yang Lebih Baik:** Karena tidak ada pemutaran 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. Umpang 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 href="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.1/jqueryui.css" rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jqueryvalidate/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	<i>required</i>	<ul style="list-style-type: none">• Memastikan bahwa field tidak kosong.• Contoh: <code>required: true</code>
2	<i>email</i>	<ul style="list-style-type: none">• Memastikan bahwa input berformat alamat email yang valid.• Contoh: <code>email: true</code>
3	<i>URL</i>	<ul style="list-style-type: none">• Memastikan bahwa input berformat URL yang valid.• Contoh: <code>url: true</code>

4	<i>date</i>	<ul style="list-style-type: none">• Memastikan bahwa input berformat tanggal yang valid (berdasarkan pengaturan regional)• Contoh: <code>date: true</code>
5	<i>dateISO</i>	<ul style="list-style-type: none">• Memastikan bahwa input berformat tanggal yang valid dalam format ISO (YYYY-MM-DD)• Contoh: <code>dateISO: true</code>



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



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

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.

```
template.blade.php <pre>
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>
```



```
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>
```

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



```
<button onclick="modalAction('{{ url('/user/create_ajax') }}')" class="btn btn-sm  
btnsuccess 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

```
<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. Selanjutkan 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
});
```

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);
}
```

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,
                    $(form).serialize(),
                    success: function(response) {
                        if(response.status){
                            $('#myModal').modal('hide');
                            Swal.fire({
                                'success',
                                'Berhasil',
                                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);
                },
                validClass: {
                    highlight: function (element, errorClass,
                    validClass) {
                        $(element).addClass('is-invalid');
                    },
                    unhighlight: function (element, errorClass,
                    validClass) {
                        $(element).removeClass('is-invalid');
                    }
                };
            });
        });
    </script>
```

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('/');
}
```

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

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]



⇒ Saat penambahan data

Tambah User

Tambah user baru

Level	Staff/Kasir
Username	Kasir 2
Nama	Kasir 2
Password

Simpan Kembali

2021 AdminLTE.io. All rights reserved. Version 3.2.0

⇒ Klik Simpan

Daftar User

Data user berhasil disimpan

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	Kasir 2	Kasir 2	Staff/Kasir	[Detail] [Edit] [Hapus]

Showing 1 to 4 of 4 entries Previous Next

2021 AdminLTE.io. All rights reserved. Version 3.2.0

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).'" class="btn btn-info
bttnsm">Detail</a> ';
        $btn .= '<a href="'.url('/user/' . $user->user_id . '/edit').'" class="btn
bttnwarning bttn-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 bttn-sm" onclick="return
confirm(\\'Apakah Anda yakin menghapus data ini?\')";>Hapus</button></form>';*/
$btn  = '<button onclick="modalAction(\'' .url('/user/' . $user->user_id .
'/show_ajax').'\')" class="btn btn-info bttn-sm">Detail</button> ';
        $btn .= '<button onclick="modalAction(\'' .url('/user/' . $user->user_id .
'/edit_ajax').'\')" class="btn btn-warning bttn-sm">Edit</button> ';
        $btn .= '<button onclick="modalAction(\'' .url('/user/' . $user->user_id .
'/delete_ajax').'\')" class="btn btn-danger bttn-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

```
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('/{id}', [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::delete('/{id}', [UserController::class, 'destroy']); // Menghapus data user
});
```

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]);
}
```

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>
@else
    <form action="{{ url('/user/' . $user->user_id . '/update_ajax') }}" method="POST"
id="formedit" @csrf
    @method('PUT')
    <div id="modal-master" class="modal-dialog modal-lg" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Edit Data User</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label>Level Pengguna</label>
                    <select name="level_id" id="level_id" class="form-control" required>
                        <option value="">- Pilih Level -</option>
                        @foreach($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 textdanger"></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 textdanger"></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="formcontrol">
                    <small class="form-text text-muted">Abaikan jika tidak ingin ubah password</small>
                    <small id="error-password" class="error-text form-text textdanger"></small>
                </div>
                <div class="modal-footer">
                    <button type="button" data-dismiss="modal" class="btn btnwarning">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,
                $(form).serialize(),
                function(response) {
                    if(response.status){
                        Swal.fire({
                            title: 'Berhasil',
                            icon: 'success',
                            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');
        }
    });
});
```

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('/');
}
```

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



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	Kasir 2	Kasir 2	Staff/Kasir	[Detail] [Edit] [Hapus]

⇒ Klik Edit

Level Pengguna: Staff/Kasir

Username: Kasir 2

Nama: Kasir 2

Abakan jika tidak ingin mengubah password.

Batal Simpan

⇒ Saat di klik simpan



The screenshot shows a web-based application interface for managing users. On the left, there's a sidebar with various menu items like Dashboard, Data Pengguna, Level User, and Data User (which is currently selected). The main content area is titled 'Daftar User' and displays a table of user data. One row in the table has been updated, and a modal dialog box is overlaid on the page, indicating success ('Berhasil') with a green checkmark icon. The message 'Data berhasil diupdate' is visible inside the modal. At the bottom right of the modal, there's an 'OK' button.

⇒ Hasilnya Nama Kasir 2 terupdate menjadi Kasir Dua

This screenshot shows the same application interface after the update. The 'Data User' table now lists four entries. The fourth entry, which previously had 'Kasir 2' as the name, now shows 'Kasir Dua'. The rest of the data remains the same: ID 1 is 'admin' with 'Administrator' level; ID 2 is 'Manager' with 'Manager' level; ID 3 is 'Staff' with 'Staff/Kasir' level; and the newly updated entry is ID 4 with 'Kasir Dua' and 'Staff/Kasir' level.

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
});
```

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]);
}
```

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>
    @else
        <form action="{{ url('/user/' . $user->user_id.'/delete_ajax') }}" method="POST"
id="formdelete" @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->nama }}</td></tr>
                    </table>
                </div>
                <div class="modal-footer">
                    <button type="button" data-dismiss="modal" class="btn btnwarning">Batal</button>
                    <button type="submit" class="btn btn-primary">Ya, Hapus</button>
                </div>
            </div>
        </form>
        <script>
            $(document).ready(function() {
$( "#form-delete" ).validate({
rules: {},
submitHandler: function(form) {
$.ajax({
```



```
url: form.action,
type: form.method,
$(form).serialize(),
function(response) {
if(response.status){
Swal.fire({
title: 'Berhasil',
icon: 'success',
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



```
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('/');
}
```

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

⇒ Tampilan Awal

The screenshot shows a web application interface titled "PWL - Starter Code". On the left, there is a sidebar with navigation links: Dashboard, Data Pengguna, Level User, Data User (which is currently selected), Data Barang, Kategori Barang, Data Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area is titled "Daftar User" and displays a table of users:

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	Kasir 2	Kasir Dua	Staff/Kasir	[Detail] [Edit] [Hapus]

At the bottom of the page, there is a footer with the text "2021 AdminLTE.io. All rights reserved." and "Version 3.2.0".

⇒ Setelah Klik Hapus



The screenshot shows a modal dialog titled 'Hapus Data User' with a yellow header bar containing a question mark icon and the text 'Konfirmasi!!! Apakah Anda ingin menghapus data seperti di bawah ini?'. Below this, it displays the user details: 'Level Pengguna : Staff/Kasir', 'Username : Kasir 2', and 'Nama : Kasir Dua'. At the bottom right of the modal are two buttons: 'Batal' (Cancel) and 'Ya, Hapus' (Yes, Delete). The background shows a table of users with 4 entries, including 'Kasir 2' at ID 4. The table has columns for 'ID', 'Username', 'Nama', and 'Level Pengguna'. The footer of the page includes the text '2021 AdminLTE.io. All rights reserved.' and 'Version 3.2.0'.

The screenshot shows a modal dialog with a green checkmark icon and the text 'Berhasil' (Successful). Below it, it says 'Data berhasil dihapus' (Data deleted successfully). At the bottom right of the modal is an 'OK' button. The background shows the same 'Daftar User' table with 3 entries remaining, excluding 'Kasir 2'. The footer of the page includes the text '2021 AdminLTE.io. All rights reserved.' and 'Version 3.2.0'.

⇒ Hasil akhir data ID 4 akan terhapus



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]

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

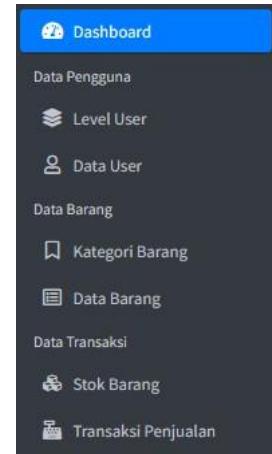
Tabel m_level

Tabel m_kategori

Tabel m_supplier

Tabel m_barang

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





1. M_level

⇒ Tampilan Awal

The screenshot shows a web-based application interface for managing user levels. The left sidebar has a dark theme with white icons and text, listing various modules like Dashboard, Data Pengguna, Level User (which is highlighted in blue), Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area has a light background. It displays a table titled 'Daftar Level User' with the following data:

ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	[Detail] [Edit] [Hapus]
2	MNG	Manager	[Detail] [Edit] [Hapus]
3	STF	Staff/Kasir	[Detail] [Edit] [Hapus]

Below the table, a message says 'Showing 1 to 3 of 3 entries'. At the bottom right of the main area, there are buttons for 'Previous' and 'Next'. The footer of the page includes the text '2021 AdminLTE.io, All rights reserved.' and 'Version 3.2.0'.

Tambah

⇒ Setelah Klik tambah

This screenshot shows the 'Tambah Level' (Add Level) page. The left sidebar is identical to the previous one. The main content area has a light background and contains a form for adding a new level. The form fields are:

- Kode Level: AST
- Level Nama: Assistance

At the bottom of the form, there are two buttons: 'Simpan' (Save) and 'Kembali' (Back). The footer of the page includes the text '2021 AdminLTE.io, All rights reserved.' and 'Version 3.2'.

⇒ Klik simpan , data akan tersimpan dengan ID 4



ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	[Detail] [Edit] [Hapus]
2	MNG	Manager	[Detail] [Edit] [Hapus]
3	STF	Staff/Kasir	[Detail] [Edit] [Hapus]
4	AST	Assistance	[Detail] [Edit] [Hapus]

Edit

⇒ Tampilan Awal

ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	[Detail] [Edit] [Hapus]
2	MNG	Manager	[Detail] [Edit] [Hapus]
3	STF	Staff/Kasir	[Detail] [Edit] [Hapus]
4	AST	Assistance	[Detail] [Edit] [Hapus]

⇒ Setelah klik edit



Daftar Level

ID	Kode Level	Nama Level	Staff/Kasir	Aksi
1	ADM			[Detail] [Edit] [Hapus]
2	MNG			[Detail] [Edit] [Hapus]
3	STF			[Detail] [Edit] [Hapus]
4	AST	Assistance Manager	Staff/Kasir	[Detail] [Edit] [Hapus]

⇒ Lalu Klik Simpan

Daftar Level

ID	Kode Level	Nama Level	Staff/Kasir	Aksi
1	ADM			[Detail] [Edit] [Hapus]
2	MNG			[Detail] [Edit] [Hapus]
3	STF			[Detail] [Edit] [Hapus]
4	AST	Assistance Manager	Staff/Kasir	[Detail] [Edit] [Hapus]

⇒ Hasil Akhir data terupdate dari Asistance menjadi Assistance Manager



ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	[Detail] [Edit] [Hapus]
2	MNG	Manager	[Detail] [Edit] [Hapus]
3	STF	Staff/Kasir	[Detail] [Edit] [Hapus]
4	ASM	Assistance Manager	[Detail] [Edit] [Hapus]

Delete

⇒ Tampilan Awal

ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	[Detail] [Edit] [Hapus]
2	MNG	Manager	[Detail] [Edit] [Hapus]
3	STF	Staff/Kasir	[Detail] [Edit] [Hapus]

⇒ Setelah klikhapus



The screenshot shows a web-based application interface for managing user levels. On the left, there's a sidebar with various menu items like Dashboard, Data Pengguna, Level User (which is selected and highlighted in blue), Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area is titled 'Daftar Level' and displays a table of existing levels:

ID	Kode Level	Nama Level
1	ADM	Staff/Kasir
2	MNG	
3	STF	
4	ASM	Assistance Manager

A modal window titled 'Hapus Data Level' is open, asking for confirmation: 'Konfirmasi !!! Apakah Anda ingin menghapus data seperti di bawah ini?'. It lists the same four levels with their details. At the bottom of the modal are two buttons: 'Batal' (Cancel) and 'Ya, Hapus' (Yes, Delete). The background table also has 'Detail', 'Edit', and 'Hapus' buttons for each row.

⇒ Setelah klik Ya,Hapus

This screenshot shows the same application after the deletion of the level with ID 4 (ASM). The table now only contains three entries: ADM, MNG, and STF. A new modal window is displayed, indicating success: 'Berhasil' (Successful) with the message 'Data berhasil dihapus' (Data was successfully deleted). There is a 'OK' button at the bottom of this modal.

⇒ Hasil Akhir Data dengan ID 4 akan terhapus



ID	Kode Level	Nama Level	Aksi
1	ADM	Administrator	[Detail] [Edit] [Hapus]
2	MNG	Manager	[Detail] [Edit] [Hapus]
3	STF	Staff/Kasir	[Detail] [Edit] [Hapus]

2. M_kategori

⇒ Tampilan Awal

ID	Kode Kategori	Nama Kategori	Aksi
1	KAT001	Pakaian	[Detail] [Edit] [Hapus]
2	KAT002	Elektronik	[Detail] [Edit] [Hapus]
3	KAT003	Makanan	[Detail] [Edit] [Hapus]
4	KAT004	Minuman	[Detail] [Edit] [Hapus]
5	KAT005	Alat Tulis	[Detail] [Edit] [Hapus]

Tambah

⇒ Setelah Klik tambah



127.0.0.1:8000/kategori/create

PWL - Starter Code

Tambah Kategori

Tambah kategori baru

Kode Kategori: BTS
The kategori.kode field must be at least 3 characters.

Kategori Nama: Buku Tulis

Simpan Kembali

2021 AdminLTE.io. All rights reserved.

Version 3.2.0

⇒ Klik Simpan

127.0.0.1:8000/kategori

PWL - Starter Code

Daftar Kategori

Daftar kategori yang terdaftar dalam sistem

Tambah Tambah Ajax

Data kategori berhasil disimpan

ID	Kode Kategori	Nama Kategori	Aksi
1	KAT001	Pakaian	[Detail] [Edit] [Hapus]
2	KAT002	Eletronik	[Detail] [Edit] [Hapus]
3	KAT003	Makanan	[Detail] [Edit] [Hapus]
4	KAT004	Minuman	[Detail] [Edit] [Hapus]
5	KAT005	Alat Tulis	[Detail] [Edit] [Hapus]
6	BTS	Buku Tulis	[Detail] [Edit] [Hapus]

Showing 1 to 6 of 6 entries

Previous 1 Next

2021 AdminLTE.io. All rights reserved.

Version 3.2.0

Edit

⇒ Setelah Klik Edit



Daftar Kategori

ID	Kode Kategori	Nama Kategori	Aksi
1	KAT001	Makanan	[Detail] [Edit] [Hapus]
2	KAT002	Minuman	[Detail] [Edit] [Hapus]
3	KAT003	Alat Tulis	[Detail] [Edit] [Hapus]
4	KAT004	Buku Tulis	[Detail] [Edit] [Hapus]
5	KAT005		[Detail] [Edit] [Hapus]
6	BTS		[Detail] [Edit] [Hapus]

⇒ Lalu Klik Simpan Data berhasil diUpdate

Daftar Kategori

ID	Kode Kategori	Nama Kategori	Aksi
1	KAT001	Makanan	[Detail] [Edit] [Hapus]
2	KAT002	Minuman	[Detail] [Edit] [Hapus]
3	KAT003	Alat Tulis	[Detail] [Edit] [Hapus]
4	KAT004	Buku Tulis	[Detail] [Edit] [Hapus]
5	KAT005		[Detail] [Edit] [Hapus]
6	BTS		[Detail] [Edit] [Hapus]



ID	Kode Kategori	Nama Kategori	Aksi
1	KAT001	Pakalan	[Detail] [Edit] [Hapus]
2	KAT002	Elektronik	[Detail] [Edit] [Hapus]
3	KAT003	Makanan	[Detail] [Edit] [Hapus]
4	KAT004	Minuman	[Detail] [Edit] [Hapus]
5	KAT005	Alat Tulis	[Detail] [Edit] [Hapus]
6	BTS	Buku Tulis Besar	[Detail] [Edit] [Hapus]

Delete

⇒ Setelah Klik Hapus

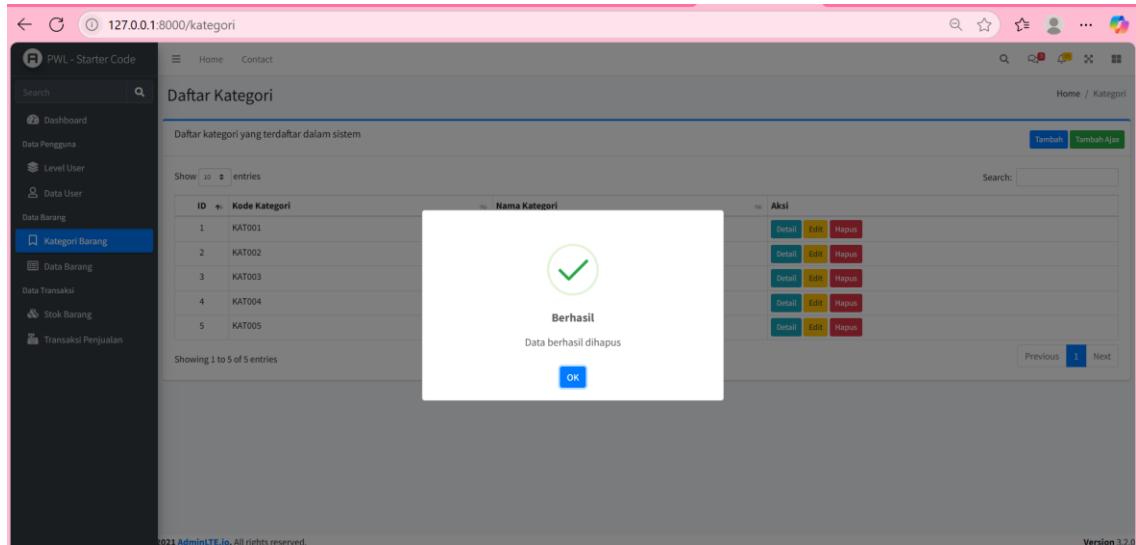
Hapus Data Kategori

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

Kode Kategori : BTS
Nama Kategori : Buku Tulis Besar

Batal Ya, Hapus

⇒ Lalu Ya,Hapus, Data Berhasil Dihapus



The screenshot shows a modal dialog box centered on the page. It contains a green checkmark icon, the word "Berhasil" (Successful), and the message "Data berhasil dihapus" (Data has been deleted successfully). There is an "OK" button at the bottom right of the modal.

ID	Kode Kategori	Nama Kategori	Aksi
1	KATO01	Pakaian	[Detail] [Edit] [Hapus]
2	KATO02	Elektronik	[Detail] [Edit] [Hapus]
3	KATO03	Makanan	[Detail] [Edit] [Hapus]
4	KATO04	Minuman	[Detail] [Edit] [Hapus]
5	KATO05	Alat Tulis	[Detail] [Edit] [Hapus]

3. M_supplier

⇒ Tampilan Awal



ID	Kode Supplier	Nama Supplier	Alamat Supplier	Aksi
1	SKT001	Skintific Official	Jakarta	[Detail, Edit, Hapus]
2	SKT002	Distributor A	Surabaya	[Detail, Edit, Hapus]
3	SKT003	Distributor B	Bandung	[Detail, Edit, Hapus]

Tambah

⇒ Setelah Klik Tambah

Kode Supplier: SKT004
Nama Supplier: Distributor C
Alamat Supplier: Malang

⇒ Lalu Klik Simpan Data akan bertambah



ID	Kode Supplier	Nama Supplier	Alamat Supplier	Aksi
1	SKT001	Skintific Official	Jakarta	[Detail, Edit, Hapus]
2	SKT002	Distributor A	Surabaya	[Detail, Edit, Hapus]
3	SKT003	Distributor B	Bandung	[Detail, Edit, Hapus]
4	SKT004	Distributor C	Malang	[Detail, Edit, Hapus]

Edit

⇒ Setelah Klik Edit

ID	Kode Supplier	Nama Supplier	Alamat Supplier
1	SKT001	Distributor B	Bandung
2	SKT002	Distributor C	Malang
3	SKT003	Distributor C	Malang
4	SKT004	Distributor C	Malang

⇒ Lalu Klik Simpan Data Berhasil Diupdate



The screenshot shows a web-based application interface for managing suppliers. On the left is a sidebar with various menu items like Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area is titled 'Daftar Supplier' and displays a table of supplier data. A modal window in the center says 'Berhasil' (Successful) with the message 'Data berhasil diupdate' (Data has been updated). At the bottom right of the modal is a blue 'OK' button.

This screenshot shows the same application interface as the previous one, but the modal window is closed. The main content area displays a table of four supplier entries:

ID	Kode Supplier	Nama Supplier	Alamat Supplier	Aksi
1	SKT001	Skintific Official	Jakarta	[Detail] [Edit] [Hapus]
2	SKT002	Distributor A	Surabaya	[Detail] [Edit] [Hapus]
3	SKT003	Distributor B	Bandung	[Detail] [Edit] [Hapus]
4	SKT004	Distributor C	Kota Malang	[Detail] [Edit] [Hapus]

At the bottom right of the table is a 'Previous' button with the number '1' and a 'Next' button.

Delete

⇒ Setelah Klik Hapus



The screenshot shows a web-based application interface for managing supplier data. On the left, there's a sidebar with various menu items like Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main area is titled 'Daftar Supplier' and lists four entries: SKT001, SKT002, SKT003, and SKT004. A modal dialog box is centered over the list, titled 'Hapus Data Supplier'. It contains a yellow header bar with the text 'Konfirmasi !!!' and 'Apakah Anda ingin menghapus data seperti di bawah ini?'. Below this, it displays the details of the selected supplier: 'Kode Supplier : SKT004', 'Nama Supplier : Distributor C', and 'Alamat Supplier : Kota Malang'. At the bottom of the modal are two buttons: 'Batal' (Cancel) and 'Ya, Hapus' (Yes, Delete). The background of the main page shows a table with columns for ID, Kode Supplier, Nama Supplier, Alamat Supplier, and Aksi (Actions).

⇒ Lalu Klik Ya,Hapus

This screenshot shows the same web application after the deletion process has been completed. The modal dialog from the previous screenshot has closed, and a new message box is displayed in the center. It features a green checkmark icon and the word 'Berhasil' (Successful). Below this, a smaller message reads 'Data berhasil dihapus' (Data has been deleted successfully). At the bottom of this message box is a single 'OK' button. The background table now only shows three entries: SKT001, SKT002, and SKT003.

⇒ Data dengan ID 4 akhir terhapus



The screenshot shows a web-based application interface for managing suppliers. The left sidebar contains navigation links for PWL - Starter Code, including Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area is titled 'Daftar Supplier' and displays a table of supplier information. The table columns are ID, Kode Supplier, Nama Supplier, Alamat Supplier, and Aksi (Actions). The data shows three entries: SKT001 (Skintific Official, Jakarta), SKT002 (Distributor A, Surabaya), and SKT003 (Distributor B, Bandung). Each row has 'Detail', 'Edit', and 'Hapus' buttons. Below the table, a message says 'Showing 1 to 3 of 3 entries'. At the bottom right, there are 'Previous' and 'Next' buttons, and a note 'Version 3.2.1'.

4. M_barang ⇒ Tampilan Awal

The screenshot shows a web-based application interface for managing products. The left sidebar contains navigation links for PWL - Starter Code, including Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area is titled 'Daftar Barang' and displays a table of product information. The table columns are ID, Kode Barang, Nama Barang, Harga Beli, Harga Jual, Kategori Barang, and Aksi. The data shows 10 entries from BRG001 to BRG010, each with its name, purchase price, selling price, category (e.g., Pakaihan, Elektronik, Makanan, Minuman, Alat Tulis), and action buttons. Below the table, a message says 'Showing 1 to 10 of 15 entries'. At the bottom right, there are 'Previous' and 'Next' buttons, and a note 'Version 3.2.0'.

Tambah



127.0.0.1:8000/barang/create

Tambah Barang

Tambah barang baru

Kategori	Pakaian
Kode Barang	BRG016
Nama Barang	Baju Tidur
Harga Beli	450000
Harga Jual	50000

Simpan **Kembali**

2021 AdminLTE.io. All rights reserved. Version 3.2.0

⇒ Lalu Klik Simpan, Data berhasil disimpan

127.0.0.1:8000/barang

Daftar Barang

Daftar barang yang terdaftar dalam sistem

Data barang berhasil disimpan

ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
11	BRG011	Serum C	30000	40000	Makanan	Detail Edit Hapus
12	BRG012	Moisturizer C	28000	38000	Minuman	Detail Edit Hapus
13	BRG013	Sunscreen C	25000	35000	Alat Tulis	Detail Edit Hapus
14	BRG014	Facial Wash C	25000	35000	Pakaian	Detail Edit Hapus
15	BRG015	Toner C	27000	37000	Elektronik	Detail Edit Hapus
16	BRG016	Baju Tidur	450000	50000	Pakaian	Detail Edit Hapus

Showing 11 to 16 of 16 entries

2021 AdminLTE.io. All rights reserved. Version 3.2.0

Edit

⇒ Setelah Klik Edit



The screenshot shows a Laravel application interface. On the left, there's a sidebar with navigation links like Home, Contact, Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Barang (which is highlighted), Data Transaksi, Stok Barang, and Transaksi Penjualan. The main content area has a header "Daftar Barang" and a sub-header "Daftar barang yang terdaftar dalam sistem". A modal window titled "Edit Data Barang" is open, showing fields for "Kategori Barang" (Pakaian), "Kode Barang" (BRG016), "Nama Barang" (Baju Tidur Anak), "Harga Beli" (450000), and "Harga Jual" (50000). Below the modal, a table lists products from ID 11 to 16. At the bottom right of the modal are "Batal" and "Simpan" buttons. To the right of the modal, there's a list of actions (Detail, Edit, Hapus) for each item, and at the bottom right, there are "Previous", "1", "2", and "Next" buttons.

127.0.0.1:8000/barang

PWL - Starter Code

Home Contact

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Barang

Data Transaksi

Stok Barang

Transaksi Penjualan

Daftar Barang

Daftar barang yang terdaftar dalam sistem

Filter: - Semua - Kategori Barang

Show 10 entries

ID	Kode Barang
11	BRG011
12	BRG012
13	BRG013
14	BRG014
15	BRG015
16	BRG016

Showing 11 to 16 of 16 entries

Edit Data Barang

Kategori Barang: Pakaian

Kode Barang: BRG016

Nama Barang: Baju Tidur Anak

Harga Beli: 450000

Harga Jual: 50000

Aksi:

Batal Simpan

Search: []

Previous 1 2 Next

2021 AdminLTE.io. All rights reserved.

Version 3.2.0

⇒ Lalu Klik Simpan, Data berhasil di Update

The screenshot shows a web-based application interface for managing inventory. On the left sidebar, there are several menu items: Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The 'Data Barang' item is currently selected and highlighted in blue.

The main content area has a title 'Daftar Barang' and a subtitle 'Daftar barang yang terdaftar dalam sistem'. Below this, there is a search bar and a filter dropdown set to 'Semua'. A modal window is open in the center, containing a large green checkmark icon and the word 'Berhasil'. Below the modal, a message says 'Data berhasil diupdate' and a blue 'OK' button is visible. The main table lists 10 items with columns for ID, Kode Barang, Nama Barang, and two numerical columns. The table rows are as follows:

ID	Kode Barang	Nama Barang		
1	BRG001	Facial Wash A		
2	BRG002	Facial Wash B		
3	BRG003	Toner A		
4	BRG004	Toner B		
5	BRG005	Serum A		
6	BRG006	Serum B		
7	BRG007	Moisturizer A	28000	38000
8	BRG008	Moisturizer B	32000	42000
9	BRG009	Sunscreen A	25000	35000
10	BRG010	Sunscreen B	27000	37000

At the bottom right of the main table, there are buttons for 'Previous' and 'Next'. To the right of the main table, there is another table titled 'Kategori Barang' with columns for 'Aksi' (Detail, Edit, Hapus) and rows for various categories like Pakalan, Elektronik, Makanan, Minuman, and Alat Tulis.



Delete

⇒ Setelah Klik Hapus

⇒ Lalu Klik Ya,Hapus, Maka data dengan ID 16 berhasil dihapus



Screenshot of a web application showing a successful deletion message.

The application title is "PWL - Starter Code". The left sidebar shows navigation links: Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The "Data Barang" link is currently selected.

The main content area is titled "Daftar Barang" and displays a table of registered items:

ID	Kode Barang	Nama Barang	
1	BRG001	Facial Wash A	
2	BRG002	Facial Wash B	
3	BRG003	Toner A	
4	BRG004	Toner B	
5	BRG005	Serum A	
6	BRG006	Serum B	
7	BRG007	Moisturizer A	28000
8	BRG008	Moisturizer B	32000
9	BRG009	Sunscreen A	25000
10	BRG010	Sunscreen B	27000

A modal window in the center says "Berhasil" (Successful) and "Data berhasil dihapus" (Data deleted successfully). A blue "OK" button is visible.

On the right side, there is a smaller table showing categories and actions:

Kategori Barang	Aksi
Pakalan	[Detail] [Edit] [Hapus]
Pakaian	[Detail] [Edit] [Hapus]
Elektronik	[Detail] [Edit] [Hapus]
Makanan	[Detail] [Edit] [Hapus]
Minuman	[Detail] [Edit] [Hapus]
Alat Tulis	[Detail] [Edit] [Hapus]

At the bottom, there are navigation links: Previous, Next, Home, and Barang.

Screenshot of the same web application showing a different set of registered items.

The application title is "PWL - Starter Code". The left sidebar shows navigation links: Dashboard, Data Pengguna, Level User, Data User, Data Barang, Kategori Barang, Data Transaksi, Stok Barang, and Transaksi Penjualan. The "Data Barang" link is currently selected.

The main content area is titled "Daftar Barang" and displays a table of registered items:

ID	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori Barang	Aksi
11	BRG011	Serum C	30000	40000	Makanan	[Detail] [Edit] [Hapus]
12	BRG012	Moisturizer C	28000	38000	Minuman	[Detail] [Edit] [Hapus]
13	BRG013	Sunscreen C	25000	35000	Alat Tulis	[Detail] [Edit] [Hapus]
14	BRG014	Facial Wash C	25000	35000	Pakalan	[Detail] [Edit] [Hapus]
15	BRG015	Toner C	27000	37000	Elektronik	[Detail] [Edit] [Hapus]

A footer note at the bottom left says "2021 AdminLTE.io, All rights reserved." and "Version 3.2.C" is mentioned at the bottom right.

*** Sekian, dan selamat belajar ***