

# PHP Admin Template

## Jagowebdev.com

### I. Pengantar

PHP Admin Template Jagowebdev dikembangkan untuk memudahkan pengembangan aplikasi. Anda tidak perlu membuat dari awal manajemen usernya, seperti bagaimana user login, hak akses pada user tersebut, dll sehingga Anda dapat fokus dalam pengembangan aplikasi.

Berikut ini beberapa hal penting yang perlu diketahui:

### II. Requirements

Aplikasi admin template ini sudah dites dan berjalan dengan baik di lingkungan pengembangan PHP versi 8 dan MariaDB versi 10.4. Untuk itu sangat disarankan menggunakan PHP dan MariaDB dengan versi tersebut. Untuk mudahnya bisa menggunakan XAMPP versi 8.0.3, untuk mendownloadnya, silakan kunjungi: <https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/>

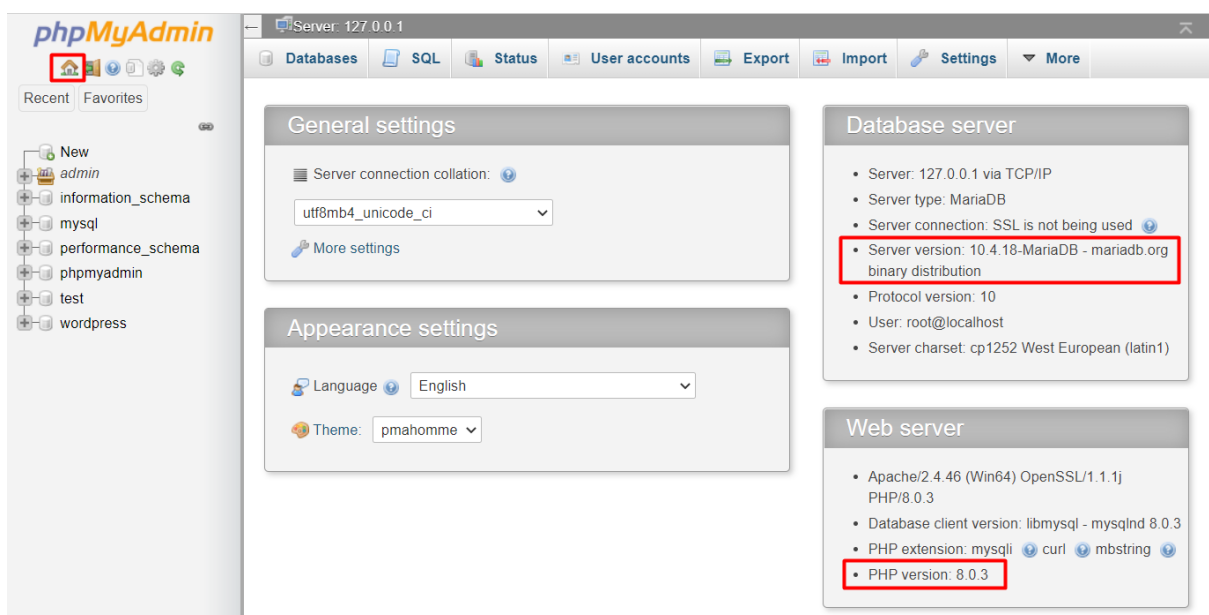
Database:

Di beberapa kasus, database tidak dapat diimpor menggunakan MySQL karena terdapat beberapa konfigurasi yang berbeda antara MariaDB dan MySQL. Kedepan, akan kami usahakan dilakukan penyesuaian agar dapat berjalan di kedua platform database tersebut.

Dibeberapa kasus juga kammi temui bahwa database tidak dapat berjalan dengan baik di MariaDB versi < 10.4. Untuk itu kami sangat menyarankan untuk menggunakan MariaDB versi 10.4 keatas.

Mengecek Server:

Untuk mengetahui versi PHP dan Database yang digunakan, Anda dapat membuka PHPMYAdmin kemudian menegcek panel sebelah kanan sebagai berikut:



Jika panek tidak seperti gambar diatas, silakan klik icon home yang ada di pojok kiri atas

### III. Install Aplikasi

Untuk install aplikasi ikuti langkah berikut:

- a. Copy file php ke folder htdocs, misal jika di copy ke folder htdocs/admin\_template maka struktur foldernya akan tampak seperti berikut:

DATA (D:) > xampp > htdocs > admin_template	
Name	Type
app	File folder
public	File folder
system	File folder
.htaccess	HTACCESS File
index.php	PHP File
LISENSI.txt	Text Document
php-admin-template-sql.sql	SQL File

- b. Buat database dengan nama yang dikehendaki, misal penjualan, selanjutnya load file php-admin-template-sql.sql yang disertakan pada file download ke database tersebut.
- c. Edit file config/config.php edit bagian BASE\_URL sesuai dengan url dimana aplikasi diinstall dan edit config/database.php sesuai dengan konfigurasi database Anda.

**Catatan:** Agar lebih aman, sebaiknya menggunakan database MariaDB bukan MySQL, contoh bundle yang menggunakan database MariaDB adalah XAMPP

### IV. Setup Awal Aplikasi

#### 1. Membuat Menu

- a. Membuat menu

Untuk membuat menu, klik menu website > Menu. Di halaman menu, klik Tambah Menu, isikan parameter, kemudian klik Submit

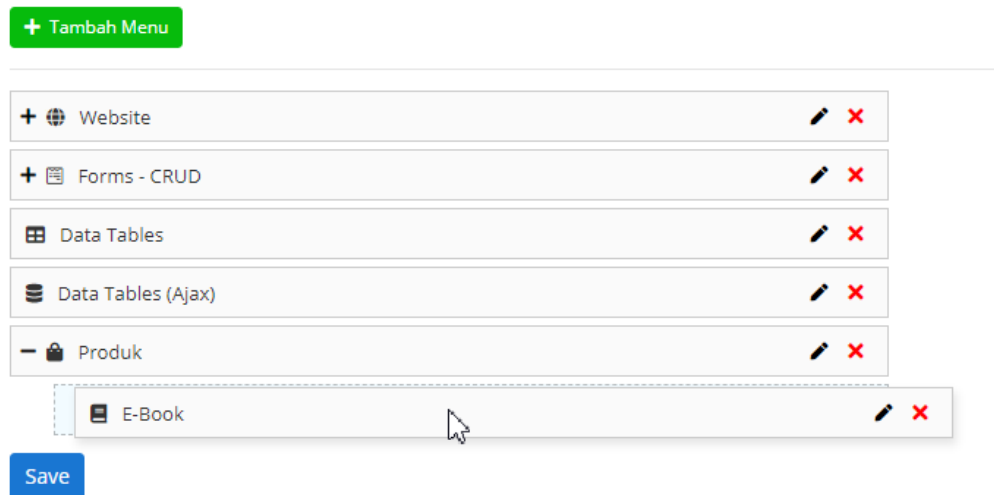
Penting diperhatikan bahwa agar nantinya menu dapat terhighlight ketika module terkait menu tersebut dibuka, maka pada bagian isian Module, pilih module yang sesuai, yang pada contoh diatas kita isi Module produk. Jika module belum dibuat, Anda dapat membuatnya terlebih dahulu pada menu Module, atau bagian Module dapat dikosongkan terlebih dahulu, kemudian mengisinya pada saat melakukan editing menu.

Menu yang dibuat akan berada di posisi paling atas pada hierarki menu, Anda dapat mengubah urutan dengan mengklik dan men drag menu yang diinginkan

Selanjutnya klik Save.

#### b. Membuat Submenu

Untuk membuat submenu, caranya, buat menu seperti langkah sebelumnya, selanjutnya geser menu yang telah dibuat tadi menjadi submenu dari menu yang diinginkan, contoh kita buat menu E-Book dan kita jadikan submenu dari menu Produk:



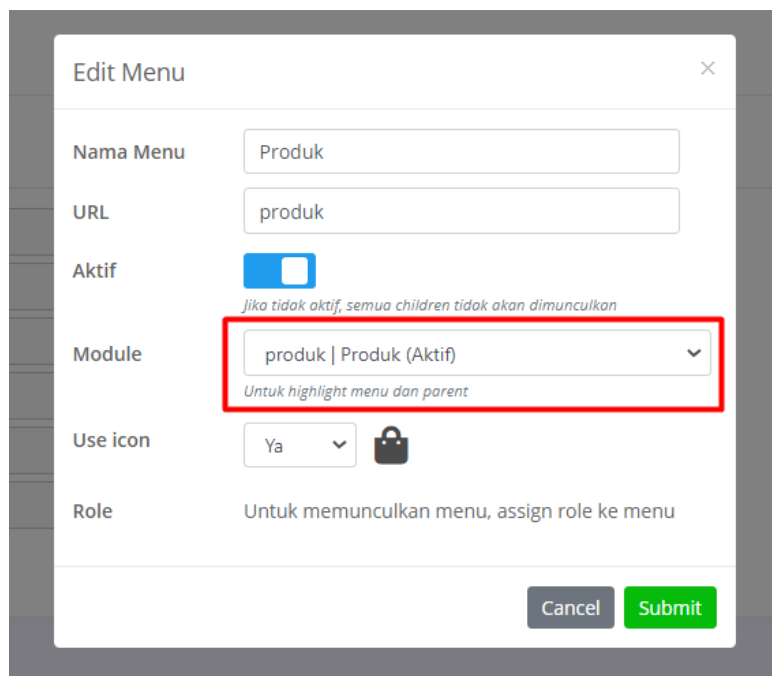
Selanjutnya klik Save.

c. Menampilkan menu

Menu tidak serta merta tampil, agar dapat tampil, menu perlu di assign terlebih dahulu ke role, untuk assign ke role, klik menu Website > Assign Role > Menu Role.

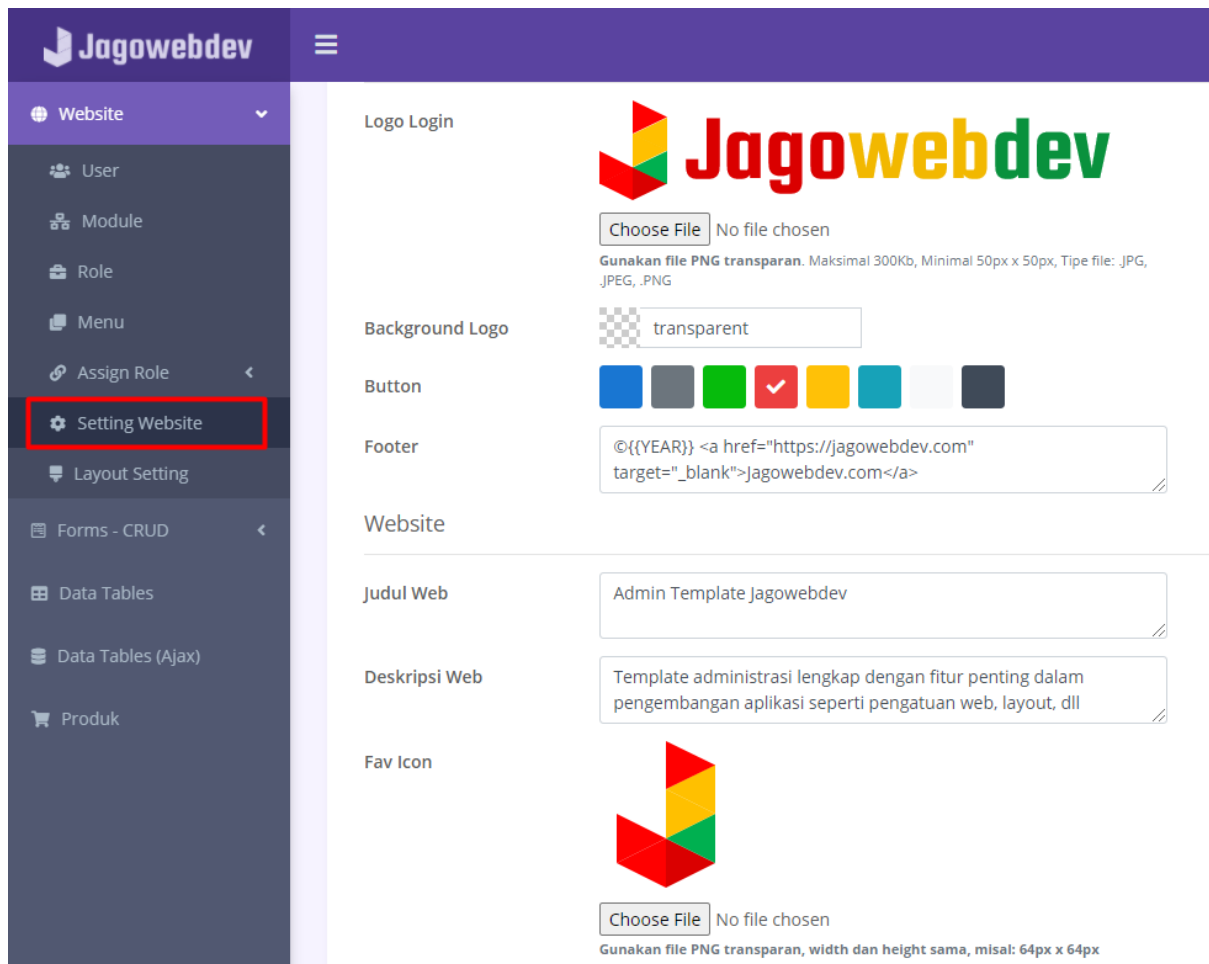
d. Menampilkan Highlight Menu ketika halaman dibuka

Sama seperti pada penjelasan membuat menu, agar menu terhighlight ketika membuka halaman tertentu yang artinya module tertentu, maka kita perlu meng-assign module ke menu tersebut, caranya edit menu yang ingin diassign modulanya kemudian pada bagian Module, pilih module yang ingin diassign ke menu tersebut



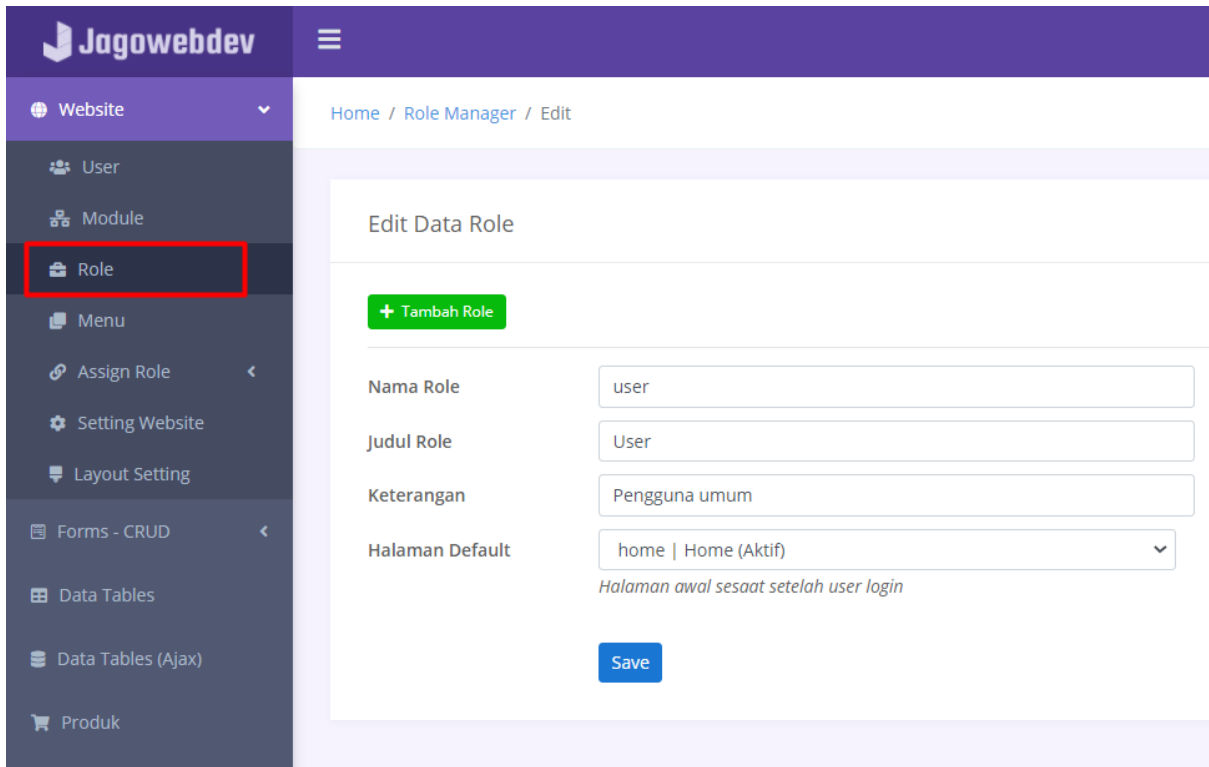
## 2. Mengubah logo

Pengaturan logo, baik logo pada favicon, halaman login, maupun halaman aplikasi dapat dilakukan melalui menu Setting Website, contoh sebagai berikut:



### 3. Membuat halaman default

Ketika user login, dapat langsung diarahkan ke halaman default sesuai dengan role user tersebut. Untuk membuat halaman default tersebut, masuk ke menu role, edit role yang ada kemudian pada halaman default pilih halaman default untuk role tersebut.



#### 4. Mode Pengembangan

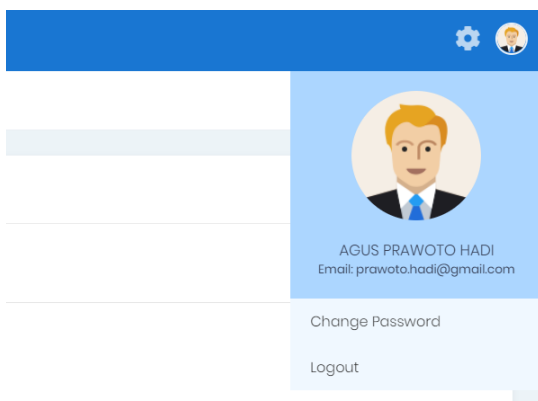
Secara default jika terjadi error, maka aplikasi akan memunculkan pesan error tersebut, Anda dapat menonaktifkan pesan error tersebut dan hanya menampilkan pesan error secara umum dengan cara mengedit file config/constant.php kemudian ganti konstanta ENVIRONMENT dari development menjadi production.

#### 5. Login

Login pertama kali untuk role admin adalah username: admin, password: admin sedangkan untuk role user adalah username: user, password: user

#### 6. Ubah Password

Untuk menjaga kerahasiaan dan keamanan password, ubah password hanya bisa dilakukan oleh user pemilik password sendiri. Caranya yaitu login ke akun, kemudian masuk ke menu Change Password yang ada di menu akun pojok kanan atas



## V. Membuat Module




### IV.1. Module Tanpa Database

Pada admin template ini, semua module disimpan di folder app/modules, semua file php terkait module disimpan di dalam folder module tersebut, misal module menu, maka semua script terkait module tersebut disimpan didalam folder app/modules/menu, misal sebagai berikut:

---

app > modules > menu >

---


Name	Type
 views	File folder
 functions.php	PHP File
 menu.php	PHP File

Untuk membuat module, pertama tama buat folder di dalam folder app/modules, misal kita akan membuat module untuk menampilkan data produk, untuk keperluan tersebut, kita buat folder dengan nama produk, di dalam folder tersebut buat file php dengan nama produk.php, misal sebagai berikut:

---

htdocs > admin\_template > app > modules > produk

---

Name	Type	Size
 produk.php	PHP File	1 KB

Misal file produk.php tersebut kita isi script sebagai berikut:

---

```
<?php
echo 'Tes Module Produk';
```

---

Selanjutnya, agar module dapat diakses, kita perlu mendaftarkan module ke sistem, caranya masuk kemenu module kemudian tambahkan module produk sebagai berikut:

**Jagowebdev**

Website ▾

User

Module

Role

Menu

Assign Role <

Setting Website

Layout Setting

Forms - CRUD <

Data Tables

Home / Module Manager / Add

### Tambah Module

[+ Tambah Module](#) [Daftar Module](#)

Nama Module  Sesuai nama yang ada di URL

Judul Module

Deskripsi

Status

[Save](#)

Isi isian pada form kemudian klik Save

Selanjutnya kita tentukan, siapa yang boleh mengakses module tersebut, caranya masuk ke menu Assign Role, sub menu Module Role seperti tampak pada gambar berikut:

**Jagowebdev**

Website ▾

User

Module

Role

Menu

Assign Role ▾

User Role

**Module Role**

Menu Role

Setting Website

Layout Setting

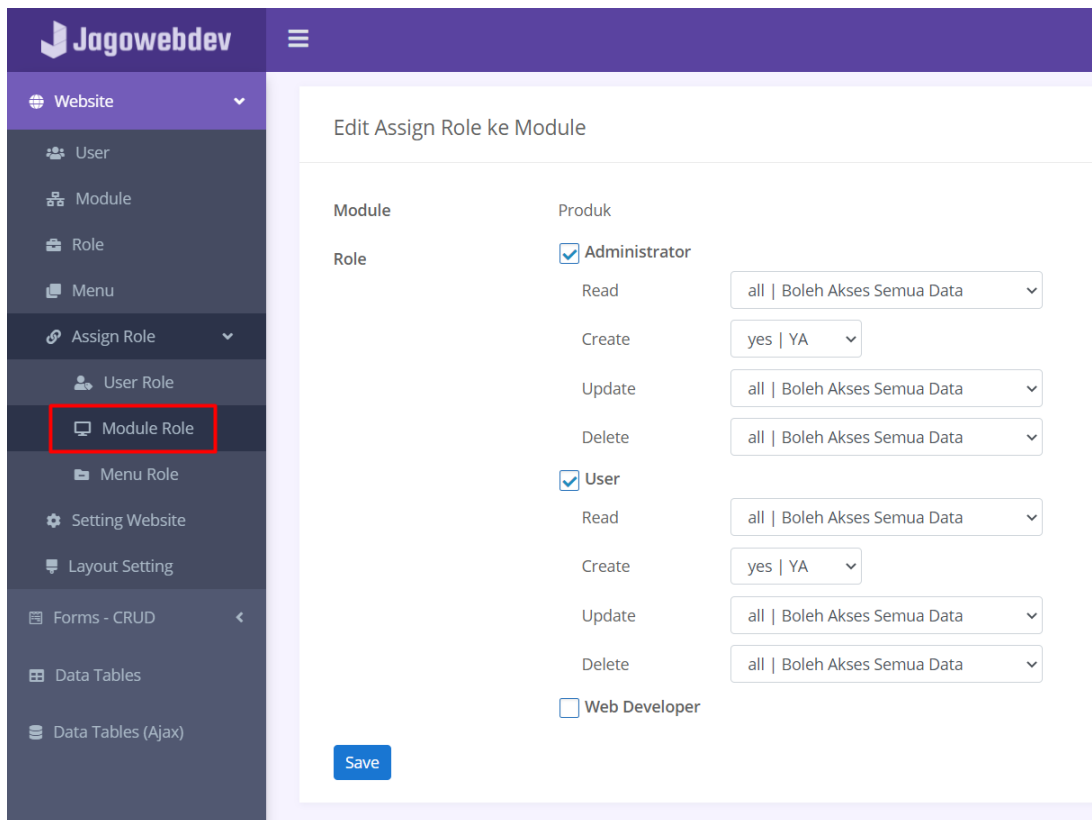
Forms - CRUD <

Data Tables

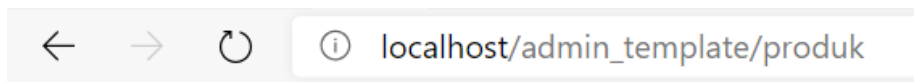
8	menu-role	Menu - Role	Administrator ✕	<a href="#">Edit</a> <a href="#">Detail</a>
9	image-upload	Image Upload	Administrator ✕ User ✕	<a href="#">Edit</a> <a href="#">Detail</a>
15	setting	Web Setting	Administrator ✕ User ✕	<a href="#">Edit</a> <a href="#">Detail</a>
16	setting-web	Setting Web	Administrator ✕	<a href="#">Edit</a> <a href="#">Detail</a>
21	options-dinamis	Options Dinamis	Administrator ✕ User ✕	<a href="#">Edit</a> <a href="#">Detail</a>
22	input-dinamis	Input Dinamis	Administrator ✕ User ✕	<a href="#">Edit</a> <a href="#">Detail</a>
25	home	Home	Administrator ✕ User ✕ Web Developer ✕	<a href="#">Edit</a> <a href="#">Detail</a>
26	multiple-fileupload	Multiple File Upload	Administrator ✕ User ✕	<a href="#">Edit</a> <a href="#">Detail</a>
27	datatables	Data Tables	Administrator ✕ User ✕ Web Developer ✕	<a href="#">Edit</a> <a href="#">Detail</a>
28	datatables-ajax	Data Tables Ajax	Administrator ✕ User ✕ Web Developer ✕	<a href="#">Edit</a> <a href="#">Detail</a>
29	produk	Produk		<a href="#">Edit</a> <a href="#">Detail</a>

Selanjutnya pilih tombol edit pada module produk. Pada halaman Edit Assign Role ke Module, pilih role yang ingin diberi akses sebagai berikut:





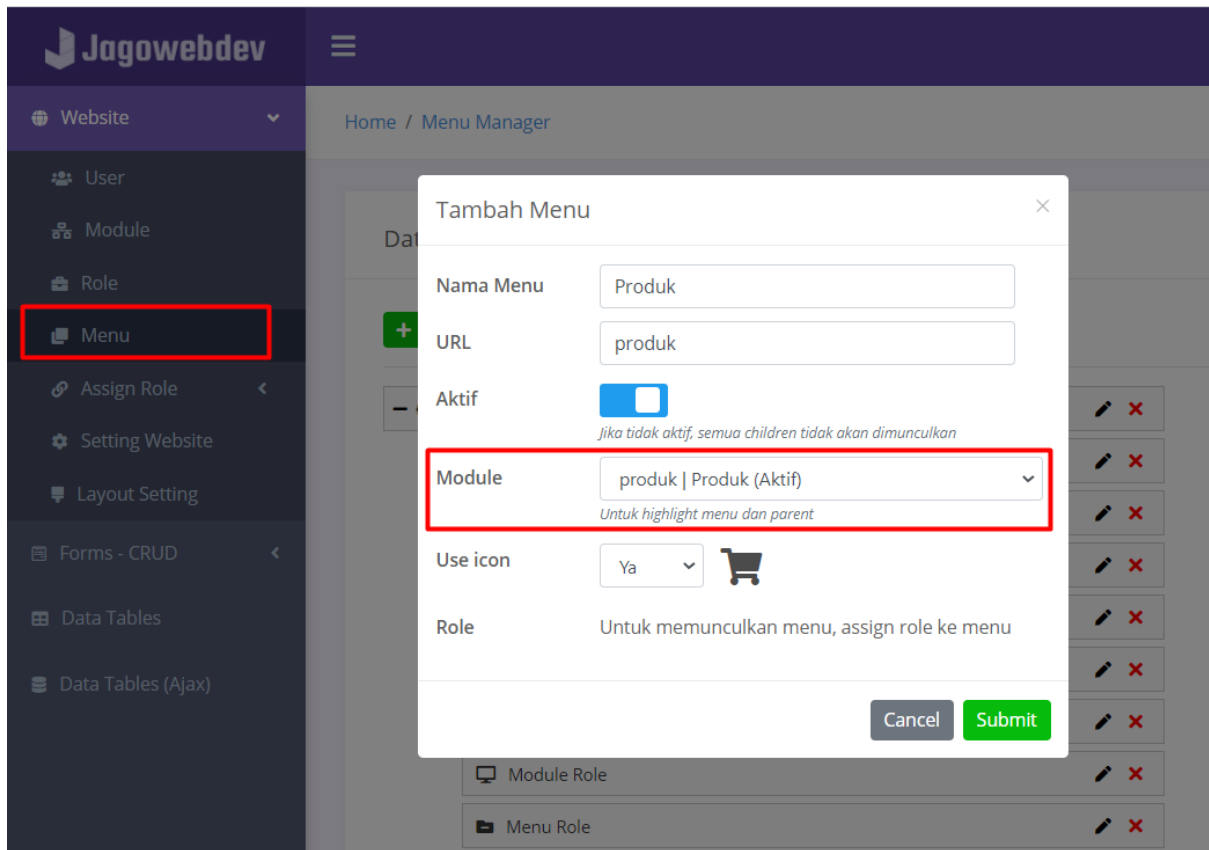
Setelah telah berhasil, maka module tersebut sudah bisa diakses, pada contoh ini, karena aplikasi saya letakkan di folder `htdocs/admin_template`, maka alamat module produk adalah [http://localhost/admin\\_template/produk](http://localhost/admin_template/produk), contoh sebagai berikut:



**Tes Module Produk**

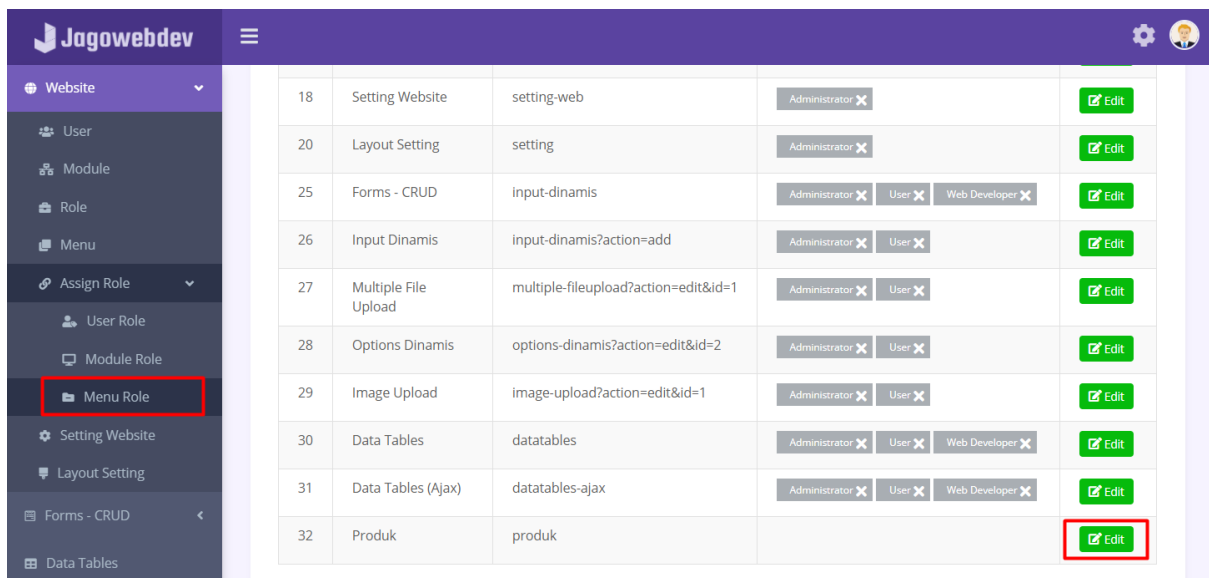
## IV.2. Membuat Menu Untuk Module

Agar module mudah diakses, maka alamat module tersebut perlu kita tambahkan pada menu, caranya, masuk ke menu "Menu" kemudian tambahkan klik Tambah Menu, pada isian menu, isikan detail menu, misal seperti contoh berikut:

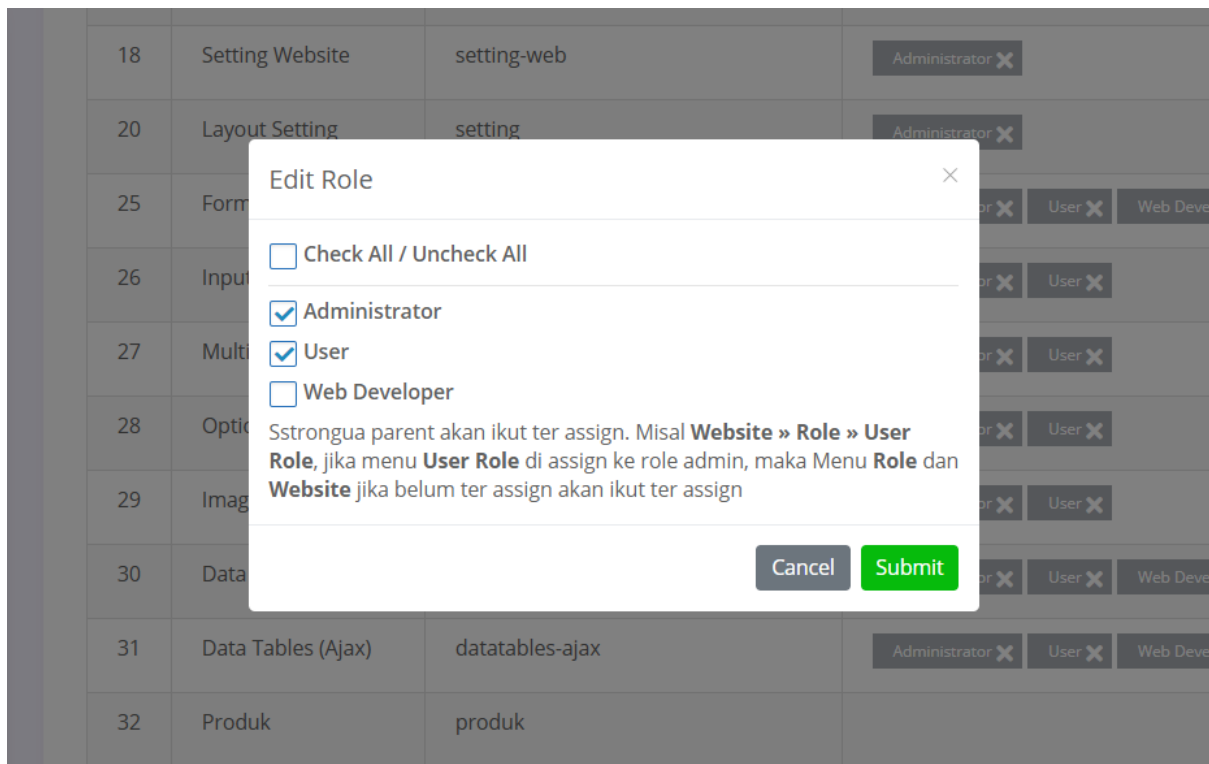


**Penting diperhatikan** bahwa pada bagian Module kita pilih module produk yang telah kita buat sebelumnya, hal ini bertujuan agar ketika module/halaman produk dibuka, menu Produk dapat ter highlight.

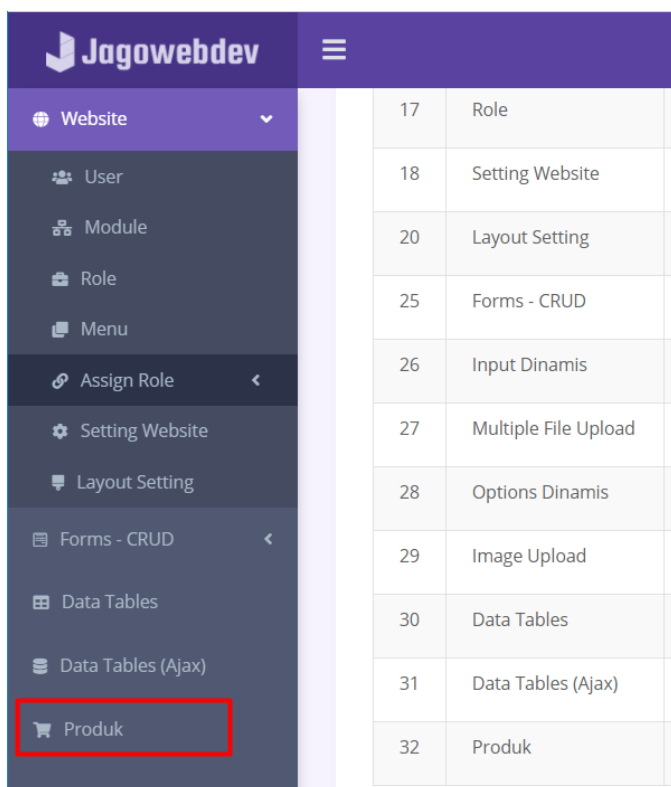
Menu yang telah kita buat tidak serta merta muncul di daftar menu, untuk memunculkannya, kita perlu meng assign menu tersebut ke role yang ada, untuk melakukannya, masuk ke menu Assign Role > Menu Role, kemudian pada menu produk, klik menu Edit sebagai berikut:



Selanjutnya pilih role yang ingin di assign dan simpan, misal seperti gambar berikut:



Setelah berhasil maka menu akan muncul disebelah kiri (jangan lupa refresh halaman)



### IV.3. Module Dengan Database

Aplikasi Admin Template sudah menyediakan class untuk melakukan query pada database, class tersebut sudah diinisiasi dan disimpan di variabel `$db` (lihat file `index.php`), untuk method method yang tersedia, dapat dilihat melalui file `system/libraries/database/pdo.php`

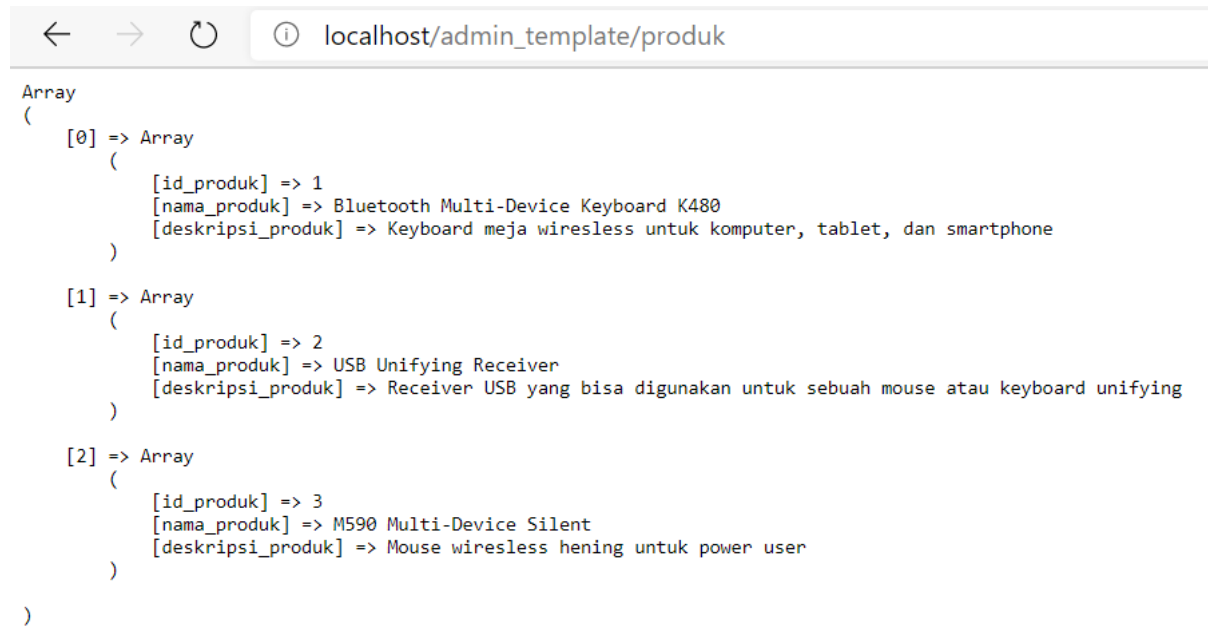
Sebagai contoh, pada module produk, file `produk.php` kita isi script sebagai berikut:

---

```
<?php
$sql = 'SELECT * FROM produk';
$result = $db->query($sql)->result();
echo '<pre>'; print_r($result);
```

---

Jika kita buka module produk, hasil yang kita peroleh adalah sebagai berikut:



Jika Anda ingin menggunakan koneksi sendiri, Anda dapat menginisiasi koneksi sendiri, sebagai contoh, file produk.php kita isi script sebagai berikut:

---

```
<?php
$sql = 'SELECT * FROM produk';
$koneksi = mysqli_connect($database['host'],
$database['username'], $database['password'],
$database['database']);
$query = mysqli_query($koneksi, $sql);
$result = mysqli_fetch_all($query, MYSQLI_ASSOC);

echo '<pre>'; print_r($result);
```

---

Jika kita jalankan, hasil yang kita peroleh sama, yaitu sebagai berikut:

```
localhost/admin_template/produk

Array
(
    [0] => Array
        (
            [id_produk] => 1
            [nama_produk] => Bluetooth Multi-Device Keyboard K480
            [deskripsi_produk] => Keyboard meja wireless untuk komputer, tablet, dan smartphone
        )

    [1] => Array
        (
            [id_produk] => 2
            [nama_produk] => USB Unifying Receiver
            [deskripsi_produk] => Receiver USB yang bisa digunakan untuk sebuah mouse atau keyboard unifying
        )

    [2] => Array
        (
            [id_produk] => 3
            [nama_produk] => M590 Multi-Device Silent
            [deskripsi_produk] => Mouse wireless hening untuk power user
        )

)
```

Agar tidak berulang membuat koneksi di setiap module yang kita buat, inisiasi koneksi (variabel \$koneksi) dapat kita letakkan di file index.php.

## IV.4. Handling Error Data Tidak Ditemukan

Jika data tidak ditemukan, Anda dapat dengan mudah memunculkan pesan error bahwa data tidak ditemukan, yaitu cukup dengan memanggil fungsi data\_notfound(). Fungsi ini ada di file system/functions.php. Sebagai contoh, file module produk.php kita ubah menjadi berikut:

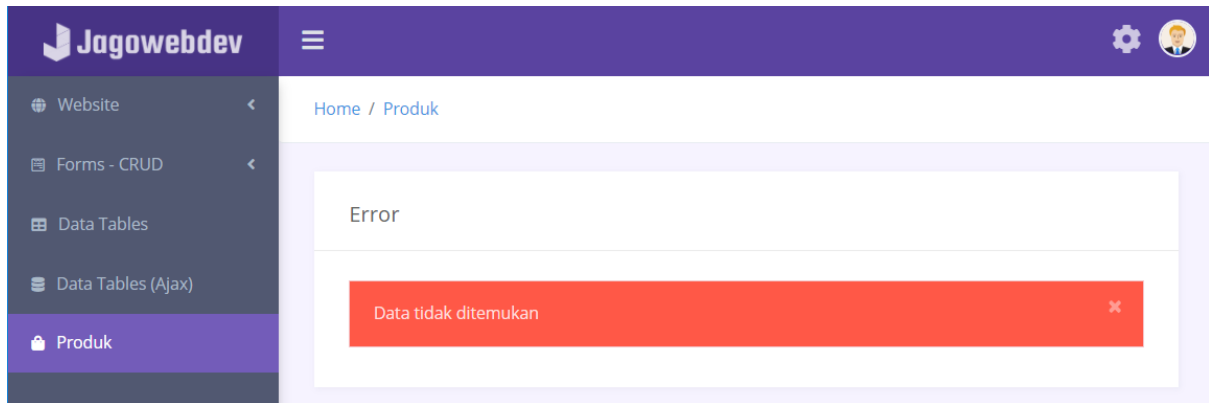
```
<?php
$koneksi    = mysqli_connect($database['host'], $database['username'],
                             $database['password'], $database['database']
                             );
$sql        = 'SELECT * FROM produk';
$query      = mysqli_query($koneksi, $sql);
$result     = mysqli_fetch_all($query, MYSQLI_ASSOC);

$data['hasil'] = $result;

if (!$data['hasil'])
    data_notfound();

load_view('view/result.php', $data);
```

Jika data produk tidak ditemukan, maka akan muncul pesan error sebagai berikut:



## IV.5. Module Dengan View

Output module yang kita bahas sebelumnya merupakan output apa adanya. Anda dapat menggunakan header dan footer bawaan aplikasi sehingga tampilan module lebih menarik, untuk menggunakannya, Anda cukup memanggil fungsi `load_view()`. Fungsi ini ada di file `system/functions.php` dan sudah otomatis terload ketika sistem berjalan.

Fungsi `load_view()` setidaknya memiliki dua parameter penting, yang pertama adalah nama file yang akan di load, parameter kedua adalah data yang akan dikirimkan ke view tersebut.

Sebagai contoh pada module produk yang telah kita buat sebelumnya kita buat file php dengan nama `result.php`. file tersebut kita masukkan kedalam folder `views`.

htdocs > admin\_template > app > modules > produk

Name	Type	Size
views	File folder	
produk.php	PHP File	1 KB

Adapun isi folder `views` adalah sebagai berikut:

htdocs > admin\_template > app > modules > produk > views

Name	Type	Size
result.php	PHP File	1 KB

Script pada file `produk.php` adalah sebagai berikut:

```
<?php
$sql      = 'SELECT * FROM produk';
$koneksi  = mysqli_connect($database['host'], $database['username'],
                           $database['password'], $database['database']
                           );
$query    = mysqli_query($koneksi, $sql);
$result   = mysqli_fetch_all($query, MYSQLI_ASSOC);

$data['hasil'] = $result;
load_view('view/result.php', $data);
```

Sedangkan pada file `views/result.php` adalah sebagai berikut:

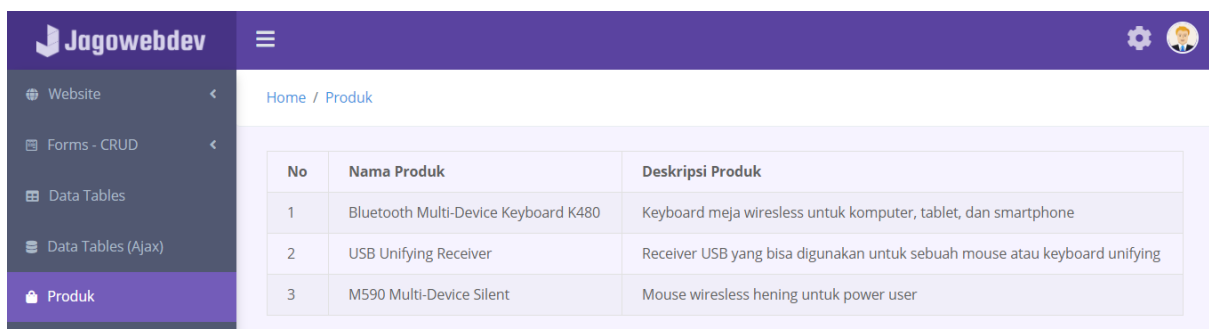
```

<table class="table table-striped table-bordered table-hover">
    <thead>
        <tr>
            <th>No</th>
            <th>Nama Produk</th>
            <th>Deskripsi Produk</th>
        </tr>
    </thead>
    <tbody>
        <?php
        $no = 1;
        foreach ($hasil as $val) {
            echo ' <tr>
                <td>' . $no . '</td>
                <td>' . $val['nama_produk'] . '</td>
                <td>' . $val['deskripsi_produk'] . '</td>
            </tr>';
            $no++;
        }
        ?>
    </tbody>
</table>

```

Perhatikan bahwa pada file produk.php, parameter kedua pada fungsi load\_view() kita isi dengan variabel \$data. Variabel \$data ini berbentuk array dengan index bernama hasil yang berisi data \$result. Ketika dikirim ke file result.php, index pada variabel data ini otomatis diekstrak menjadi variabel, sehingga pada file result.php, variabel \$hasil berisi data \$result.

Jika kita jalankan module produk, hasil yang kita peroleh adalah sebagai berikut:



No	Nama Produk	Deskripsi Produk
1	Bluetooth Multi-Device Keyboard K480	Keyboard meja wireless untuk komputer, tablet, dan smartphone
2	USB Unifying Receiver	Receiver USB yang bisa digunakan untuk sebuah mouse atau keyboard unifying
3	M590 Multi-Device Silent	Mouse wireless hening untuk power user

Perhatikan bahwa breadcrumb yang muncul adalah Home / Produk, menyesuaikan dengan nama module yang didaftarkan

Agar tampilan lebih menarik, kita tambahkan beberapa elemen dan style pada output diatas sebagai berikut:

```

<div class="card">
    <div class="card-header">
        <h5 class="card-title"><?=$current_module['judul_module']?></h5>
    </div>

    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-striped table-bordered table-hover">
                <thead>
                    <tr>
                        <th>No</th>
                        <th>Nama Produk</th>

```

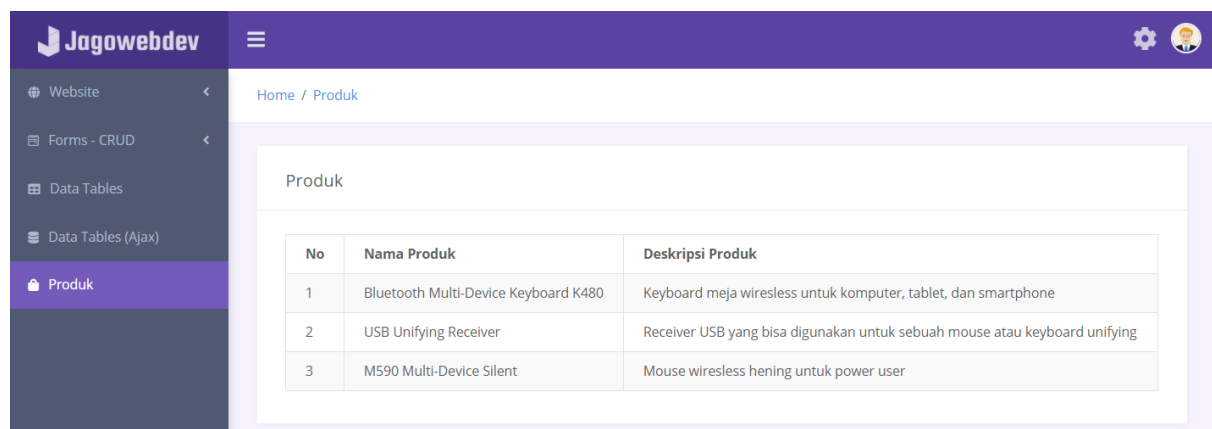
```

        <th>Deskripsi Produk</th>
    </tr>
</thead>
<tbody>
<?php
$no = 1;
foreach ($result as $val) {
    echo '<tr>
        <td>' . $no . '</td>
        <td>' . $val['nama_produk'] . '</td>
        <td>' . $val['deskripsi_produk'] . '</td>
    </tr>';
    $no++;
}
?>
</tbody>
</table>
</div>
</div>
</div>

```

Class CSS: card, card-header, card-body merupakan style bawaan aplikasi admin template, anda dapat memodifikasi style ini pada file public/themes/modern/builtin/css/site.css

Output yang dihasilkan adalah sebagai berikut:



## Aplikasi satu file

Dengan menggunakan fungsi load\_view(), seperti yang telah kita gunakan sebelumnya, akan otomatis ter load file header.php dan footer.php. Dengan cara ini, file view terpisah dari script utama. Jika Anda menginginkan file script menjadi satu file, anda dapat secara manual me load file header.php dan footer.php

Sebagai contoh, script pada file produk.php kita jadikan menjadi satu file, maka script yang kita tulis menjadi:

```

<?php
$koneksi = mysqli_connect($database['host'], $database['username'],
                        $database['password'], $database['database']
                    );

$sql = 'SELECT * FROM produk';
$koneksi = mysqli_connect($database['host'], $database['username'],
                        $database['password'], $database['database']);
$query = mysqli_query($koneksi, $sql);
$result = mysqli_fetch_all($query, MYSQLI_ASSOC);

```



---

```

include 'app/themes/modern/header.php';
?>
<table class="table table-striped table-bordered table-hover">
    <thead>
        <tr>
            <th>No</th>
            <th>Nama Produk</th>
            <th>Deskripsi Produk</th>
        </tr>
    </thead>
    <tbody>
    <?php
    $no = 1;
    foreach ($result as $val) {
        echo ' <tr>
            <td>' . $no . '</td>
            <td>' . $val['nama_produk'] . '</td>
            <td>' . $val['deskripsi_produk'] . '</td>
        </tr>';
        $no++;
    }
    ?>
    </tbody>
</table>
<?php
include 'app/themes/modern/footer.php';

```

---

## IV.6. Edit, Delete, dan Tambah Data

Bagian ini akan membahas bagaimana membuat script untuk edit, delete, dan tambah data.

### IV.6.1. Menambahkan tombol edit dan delete

Pertama tama mari kita bahas cara membuat tombol untuk edit dan delete data.

Untuk menambahkan tombol edit dan delete pada tabel data yang ditampilkan, kita dapat menggunakan fungsi `btn_action()`, fungsi ini ada di file `app/helpers/html_helper.php`, untuk menyertakan file tersebut, kita cukup menjalankan perintah `helper('html')`

Sebagai contoh script table pada file `views/result.php` pada module produk kita ubah menjadi berikut:

---

```

<table class="table table-striped table-bordered table-hover">
    <thead>
        <tr>
            <th>No</th>
            <th>Nama Produk</th>
            <th>Deskripsi Produk</th>
            <th>Aksi</th>
        </tr>
    </thead>
    <tbody>
    <?php
    helper('html');
    $no = 1;
    foreach ($result as $val) {

```

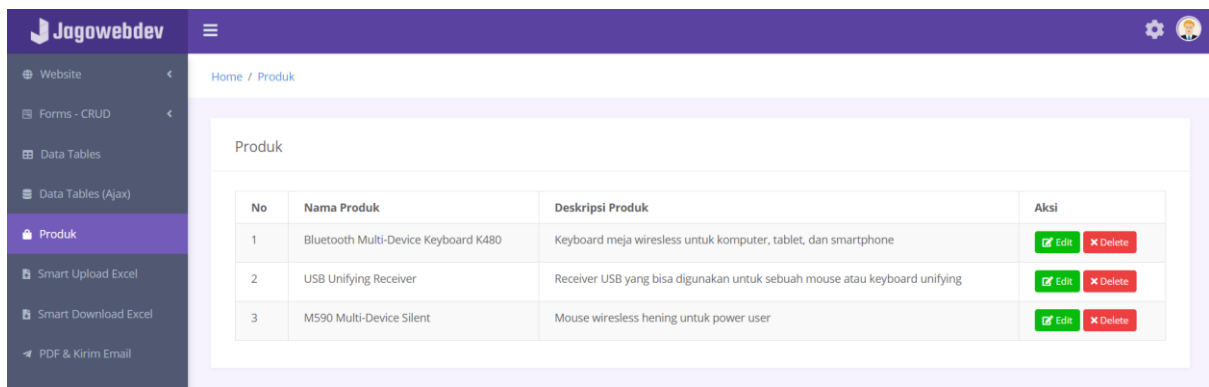
---

```

        echo
        ' <tr>
            <td>' . $no . ' </td>
            <td>' . $val['nama_produk'] . ' </td>
            <td>' . $val['deskripsi_produk'] . ' </td>
            <td>' . btn_action([
                'edit' => ['url' => '/edit?id=' .
$val['id_produk']]
                , 'delete' => ['url' => ''
                    , 'id' => $val['id_produk']
                    , 'delete-title' => 'Hapus data
produk: <strong>' . $val['nama_produk'] . ' </strong> ?'
                    ]
            ]) .
            ' </td>
        ' </tr>';
        $no++;
    }
    ?>
</tbody>
</table>

```

Jika kita jalankan, hasil yang kita peroleh adalah sebagai berikut:



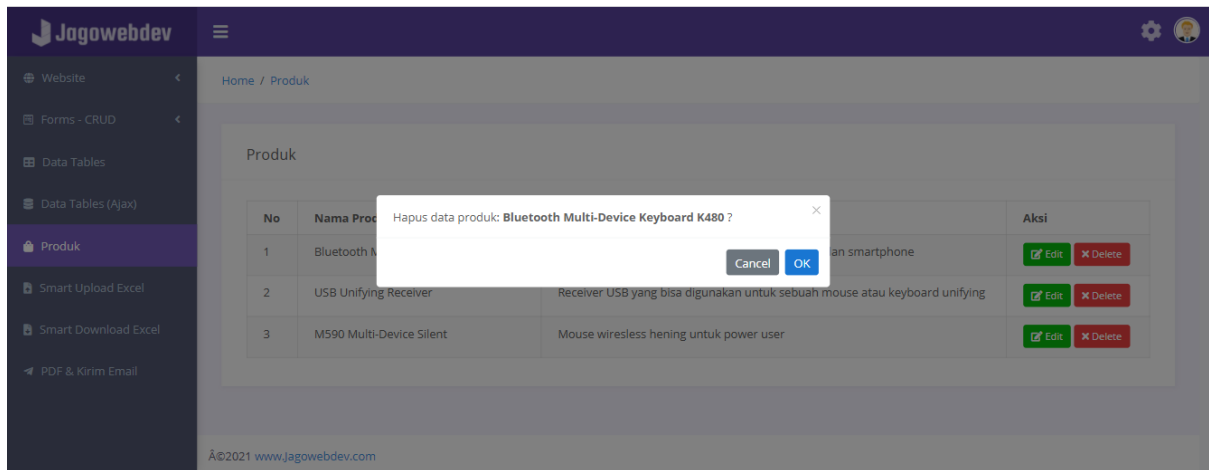
Fungsi btn\_action() diatas akan menghasilkan HTML:

```

<a href="http://localhost/admin_template/produk/edit?id=1" class="btn btn-success btn-xs mr-1">
    <span class="btn-label-icon"><i class="fa fa-edit pr-1"></i></span> Edit
</a>
<form method="post" action="">
    <button type="submit" data-action="delete-data" data-delete-title="Hapus data
produk: <strong>Bluetooth Multi-Device Keyboard K480</strong> ?" class="btn btn-danger
btn-xs">
        <span class="btn-label-icon"><i class="fa fa-times pr-1"></i></span> Delete
    </button>
    <input type="hidden" name="delete" value="delete">
    <input type="hidden" name="id" value="1">
</form>

```

Ketika button delete di klik maka akan muncul konfirmasi penghapusan data sebagai berikut:



Konfirmasi ini muncul otomatis, karena terdapat script global yang otomatis menangkap event ketika button dengan atribut `data-action="delete-data"` di klik. Script ini ada di file `public/themes/modern/js/site.js`. Adapun scriptnya adalah sebagai berikut:

```

$('table').delegate('[data-action="delete-data"]', 'click', function(e){
    e.preventDefault();
    var $this = $(this)
        , $form = $this.parents('form:eq(0)');
    bootbox.confirm({
        message: $this.attr('data-delete-title'),
        callback: function(confirmed) {
            if (confirmed) {
                $form.submit();
            }
        }
    });
});

```

## IV.6.2. Edit Data

Selanjutnya mari kita bahas cara membuat script untuk edit data.

### IV.6.2.1. Menambahkan Halaman Edit

Untuk menambahkan halaman edit, pada statement switch case yang ada pada script module kita tambahkan case 'edit'. Sebagai contoh pada file `produk.php` kita ubah script menjadi sebagai berikut:

```

<?php
$koneksi = mysqli_connect($database['host'], $database['username'],
                        $database['password'], $database['database']
                    );

switch ($_GET['action'])
{
    default:
        action_notfound();

    // INDEX
    case 'index':
        $sql = 'SELECT * FROM produk';
        $query = mysqli_query($koneksi, $sql);

```

---

```

        $result      = mysqli_fetch_all($query, MYSQLI_ASSOC);

        $data['hasil'] = $result;

        if (!$data['hasil'])
            data_notfound($data);

        load_view('views/result.php', $data);

    case 'edit':

        if (empty($_GET['id']))
            data_notfound();

        $sql = 'SELECT * FROM produk WHERE id_produk = ?';
        $result = $db->query($sql, $_GET['id'])->getJSONArray();

        if (!$result)
            data_notfound();

        $data['title'] = 'Edit Data Produk';
        $data['produk'] = $result;
        load_view('views/form.php', $data);
    }

```

---

Untuk Url default edit data nya adalah: [http://localhost/admin\\_template/produk/edit?id=1](http://localhost/admin_template/produk/edit?id=1). Selanjutnya, pada case edit, kita lakukan beberapa validasi request untuk mengantisipasi error ketika halaman edit dibuka. Pertama kita cek apakah ada id pada variabel \$\_GET

---

```

if (empty($_GET['id']))
    data_notfound($data);

```

---

Script ini untuk mengantisipasi jika ada user yang menuliskan url tanpa menyertakan id, misal [http://localhost/admin\\_template/produk/edit](http://localhost/admin_template/produk/edit) lebih lanjut mengenai fungsi data\_notfound() dapat dibaca pada sub bab Handling Error Data Tidak Ditemukan

Selanjutnya, kita juga melakukan pengecekan apakah data yang di edit tersedia, script yang digunakan adalah:

---

```

$result = $db->query($sql, $_GET['id'])->getJSONArray();
if (!$result)
    data_notfound($data);

```

---

Hal ini untuk mengantisipasi user memasukkan sembarang id, misal di database produk, id hanya ada 3, namun bisa jadi user mengubah id pada url menjadi misal 5 [http://localhost/admin\\_template/produk/edit?id=5](http://localhost/admin_template/produk/edit?id=5)

Setelah melakukan validasi request, kita load file form.php dengan fungsi `load_view('views/form.php', $data);`

Adapun script pada file form.php adalah sebagai berikut:

---

```

<div class="card">
    <div class="card-header">
        <h5 class="card-title"><?=$title?></h5>

```

---

---

```

        </div>
        <div class="card-body">
            <form method="post" action="{current_url(true)}" class="form-horizontal">
                <div class="form-group row">
                    <label class="col-sm-3 col-md-2 col-lg-3 col-xl-2 col-form-label">Nama Produk</label>
                    <div class="col-sm-5">
                        <input class="form-control" type="text"
name="nama_produk" value="{set_value('nama_produk', @$produk['nama_produk'])}"
required="required"/>
                    </div>
                </div>
                <div class="form-group row">
                    <label class="col-sm-3 col-md-2 col-lg-3 col-xl-2 col-form-label">Deskripsi Produk</label>
                    <div class="col-sm-5">
                        <textarea class="form-control"
name="deskripsi_produk">{set_value('deskripsi_produk',
@$produk['deskripsi_produk'])}</textarea>
                    </div>
                </div>
                <div class="form-group row mb-0">
                    <div class="col-sm-5">
                        <button type="submit" name="submit" value="submit"
class="btn btn-primary">Submit</button>
                        <input type="hidden" name="id"
value="{$_GET['id']}" />
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>

```

---

Pada script diatas, terdapat fungsi `current_url(true)` fungsi ini (dengan nilai argumen `true`) akan menghasilkan url beserta query string yang ada, sehingga ketika halaman form tersebut dibuka atribut `action` akan bernilai [http://localhost/admin\\_template/produk/edit?id=1](http://localhost/admin_template/produk/edit?id=1) sebagai berikut:

---

```

<form method="post" action="http://localhost/admin_template/produk/edit?id=1"
class="form-horizontal" enctype="multipart/form-data">

```

---

Selanjutnya, pada script diatas, terdapat fungsi `set_value()` sebagai berikut:

---

```

<input class="form-control" type="text" name="nama_produk"
value="{set_value('nama_produk', @$produk['nama_produk'])}"
required="required"/>

```

---

Fungsi `set_value()` ini ada di file `system/functions.php` argumen pertama dari fungsi ini adalah index dari variabel `$_POST`, sedangkan argumen kedua berisi nilai yang akan digunakan ketika index pada variabel `$_POST` tidak ditemukan, pada contoh diatas, jika data `$_POST['nama_produk']` tidak ditemukan, maka fungsi `set_value` akan mengambil nilai pada variabel `$produk['nama_produk']`.

Ketika halaman edit dibuka, maka hasil yang kita peroleh adalah:

Pada contoh diatas, form akan otomatis terisi data produk yang ingin diedit.

Selanjutnya, ketika data kita ubah, misal Nama Produk menjadi Bluetooth Multi-Device Keyboard dan kita klik submit, maka Nama Produk akan tetap Bluetooth Multi-Device Keyboard bukan Bluetooth Multi-Device Keyboard K480, hal ini karena nilai `$_POST['nama_produk']` berisi nilai Bluetooth Multi-Device Keyboard sehingga fungsi `set_value('nama_produk', @$_produk['nama_produk'])` menghasilkan nilai Bluetooth Multi-Device Keyboard

#### IV.6.2.2. Submit Data

Selanjutnya kita buat script untuk menyimpan data.

Ketika form edit produk kita submit, maka data akan dikirim melalui method post dan oleh PHP disimpan pada variabel `$_POST`

Untuk menyimpan data form yang disubmit, kita ubah script case edit menjadi berikut:

```
case 'edit':

    if (empty($_GET['id']))
        data_notfound();

    $message = [];
    if (!empty($_POST['submit'])) {
        $error = validate_form();
        if ($error) {
            $message['status'] = 'error';
            $message['message'] = $error;
        } else {
            $data_db['nama_produk'] = $_POST['nama_produk'];
            $data_db['deskripsi_produk'] = $_POST['deskripsi_produk'];
            $query = $db->update('produk', $data_db, ['id_produk' =>
$_POST['id']]);
            if ($query) {
                $message['status'] = 'ok';
                $message['message'] = 'Data berhasil disimpan';
            } else {
                $message['status'] = 'error';
                $message['message'] = 'Data gagal disimpan';
            }
        }
    }
}
```

---

```

    }
}

$sql = 'SELECT * FROM produk WHERE id_produk = ?';
$produk = $db->query($sql, $_GET['id'])->getJSONArray();

if (!$produk)
    data_notfound();

$data['title'] = 'Edit Data Produk';
$data['produk'] = $produk;
$data['message'] = $message;
load_view('views/form.php', $data);

```

---

Pada script diatas, jika dibandingkan dengan script case edit sebelumnya, terdapat tambahan script yaitu:

---

```

$message = [];
if (!empty($_POST['submit'])) {
    $error = validate_form();
    if ($error) {
        $message['status'] = 'error';
        $message['message'] = $error;
    } else {
        $data_db['nama_produk'] = $_POST['nama_produk'];
        $data_db['deskripsi_produk'] = $_POST['deskripsi_produk'];
        $query = $db->update('produk', $data_db, ['id_produk' =>
$_POST['id']]);
        if ($query) {
            $message['status'] = 'ok';
            $message['message'] = 'Data berhasil disimpan';
        } else {
            $message['status'] = 'error';
            $message['message'] = 'Data gagal disimpan';
        }
    }
}

```

---

Pada script diatas, pertama tama kita cek apakah variabel `$_POST['submit']` tidak kosong, jika ya, maka pertama tama kita panggil fungsi `validate_form()` untuk memvalidasi inputan yang ada, fungsi ini ada di bagian paling bawah setelah statement case, adapun scriptnya adalah sebagai berikut:

---

```

function validate_form() {
    $error = false;
    if (empty(trim($_POST['nama_produk']))) {
        $error[] = 'Nama Produk harus diisi';
    }

    if (empty(trim($_POST['deskripsi_produk']))) {
        $error[] = 'Deskripsi Produk harus diisi';
    }

    return $error;
}

```

---

Pada script diatas, jika variabel `$_POST['nama_produk']` dan atau variabel `$_POST['deskripsi_produk']` kosong, maka variabel `$error` akan berisi pesan error.

Kembali ke script sebelumnya, jika hasil `validate_form()` bernilai false, yang artinya tidak terdapat error, maka data disimpan (diupdate) menggunakan perintah `$db->update('produk', $data_db, ['id_produk' => $_POST['id']]);` script tersebut akan menghasilkan perintah SQL:

---

```
UPDATE produk SET nama_produk = ?,deskripsi_produk = ? WHERE id_produk = ?
```

---

Nama kolom `nama_produk` dan `deskripsi_produk`, diambil dari index pada variabel `$data_db` dan nilai kolom (pada SQL diatas berupa tanda tanya ?) juga diambil dari nilai variabel `$data_db` yang telah kita definisikan sebelumnya, sehingga kolom `nama_produk` akan berisi nilai `$_POST['nama_produk']` dan `deskripsi_produk` akan bernilai `$_POST['deskripsi_produk']`.

### IV.6.3. Delete Data

Setelah membuat script edit data, kita lanjutkan dengan membuat script delete data.

Seperti telah dibahas sebelumnya, ketika kita klik tombol delete, maka akan muncul popup konfirmasi apakah kita akan menghapus data. Jika kita klik tombol OK, maka form delete data akan tersubmit. Ketika form tersebut tersubit, maka data form delete akan dikirim menggunakan method post.

Selanjutnya untuk melakukan delete data, pada module produk.php bagian case index kita buat script sebagai berikut:

---

```
case 'index':

    if (!empty($_POST['delete'])) {
        $delete = $this->db->delete('produk', ['id_produk' => $_POST['id']]);
        if ($delete) {
            $message['status'] = 'ok';
            $message['message'] = 'Data berhasil dihapus';
        } else {
            $message['status'] = 'error';
            $message['message'] = 'Data gagal dihapus';
        }
    }

    $sql = 'SELECT * FROM produk';
    $query = mysqli_query($koneksi, $sql);
    $result = mysqli_fetch_all($query, MYSQLI_ASSOC);

    $data['hasil'] = $result;

    if (!$data['hasil'])
        data_notfound($data);

    load_view('views/result.php', $data);
```

---

Pada script diatas, jika `$_POST['delete']` didefinisikan, yang artinya ada form delete yang disubmit, maka kita jalankan perintah `$this->db->delete('produk', ['id_produk' => $_POST['id']]);` Script tersebut akan menghasilkan perintah SQL sebagai berikut:

---

```
DELETE FROM produk WHERE id_produk = 1
```

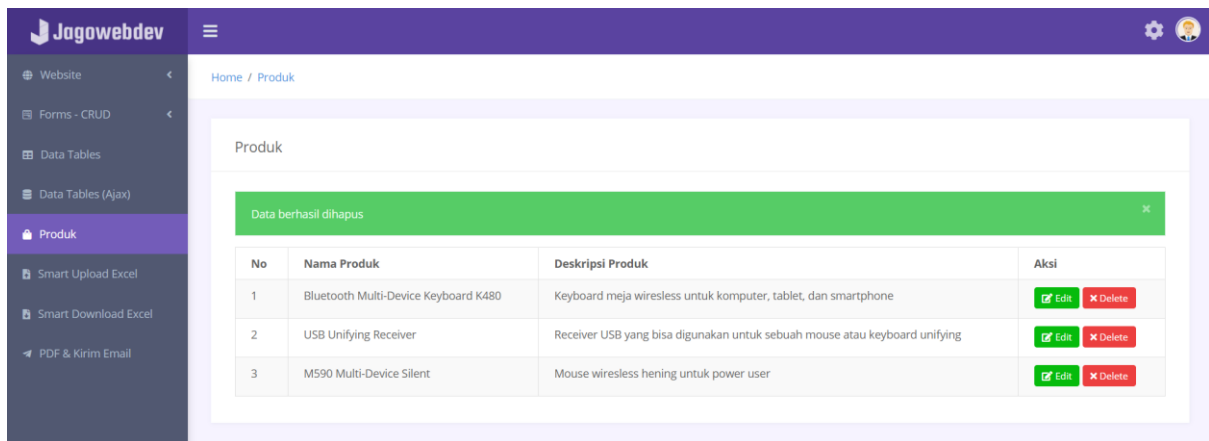
---



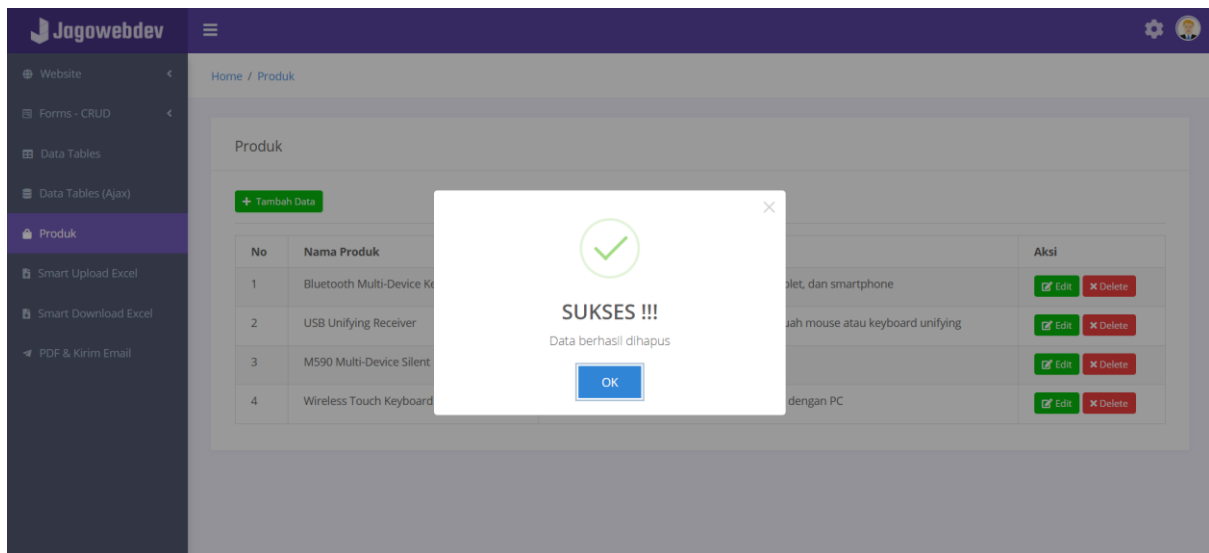
Selanjutnya, untuk menampilkan pesan hasil proses delete data, kita ubah script pada views/result.php bagian `<div class="card-body">` menjadi sebagai berikut:

```
<div class="card-body">
<?php
    if (!empty($message)) {
        show_message($message);
    } ?>
<table class="table table-striped table-bordered table-hover">
```

Jika query berhasil dijalankan, maka akan muncul pesan Data berhasil dihapus



Untuk menampilkan pesan, kita juga dapat menggunakan alert popup. Secara default, ketika aplikasi dijalankan, aplikasi meload library SweetAlert2 (disertakan di script header app/themes/modern/header.php) sehingga kita tinggal menggunakannya saja. Untuk menggunakan alert popup kita tinggal panggil dengan fungsi `show_message()`, sebagai contoh pada file `views/result.php`, fungsi `show_message()` kita ganti dengan `show_alert()`. Selanjutnya ketika kita menghapus data, maka pesan peringatannya berganti menjadi pop up sebagai berikut:



## IV.6.4. Tambah Data

Selanjutnya mari kita buat script untuk tambah data.

### F.4.1. Menambahkan tombol Tambah Data

Pertama tama, mari kita buat script untuk menampilkan tombol data. Caranya, pada script views/result.php kita tambahkan script link tambah data sebagai berikut:

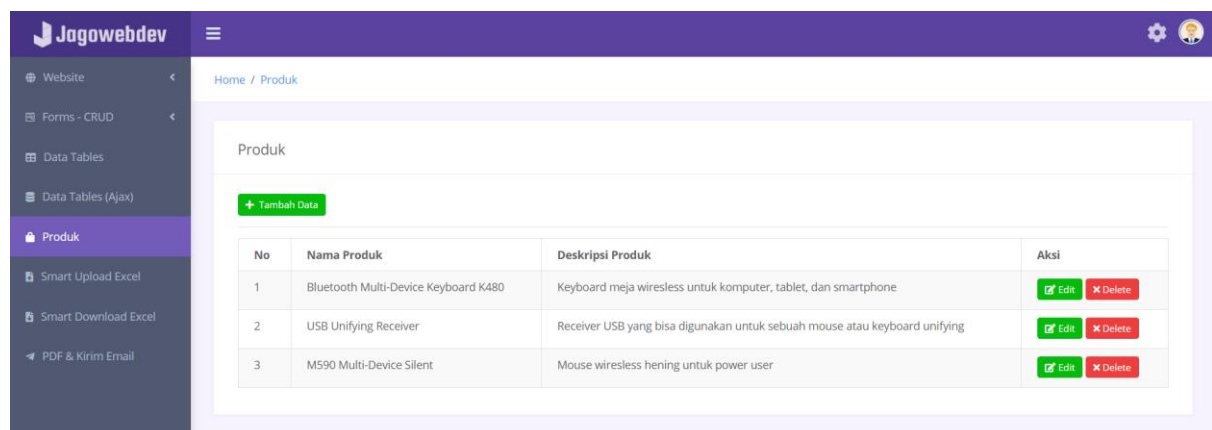
```
<div class="card-body">
<?php
    if (!empty($message)) {
        show_message($message);
    } ?>
<a href="/add" class="btn btn-success btn-xs"><i class="fas fa-plus pr-1"></i>
Tambah Data</a>
<hr/>
<table class="table table-striped table-bordered table-hover">
```

Pada script diatas, script untuk menampilkan tombol tambah data adalah sebagai berikut:

```
<a href="<?=module_url()>/add" class="btn btn-success btn-xs"><i class="fas fa-
plus pr-1"></i> Tambah Data</a>
<hr/>
```

Keterangan: pada script diatas, terdapat fungsi module\_url() fungsi ini akan menghasilkan url sesuai dengan module yang sedang di buka, pada contoh ini url yang dihasilkan adalah http://localhost/produk.

Ketika halaman produk dibuka, akan muncul tombol + Tambah Data sebagai berikut:



Selanjutnya pada script produk.php, kita tambahkan case add sebagai berikut:

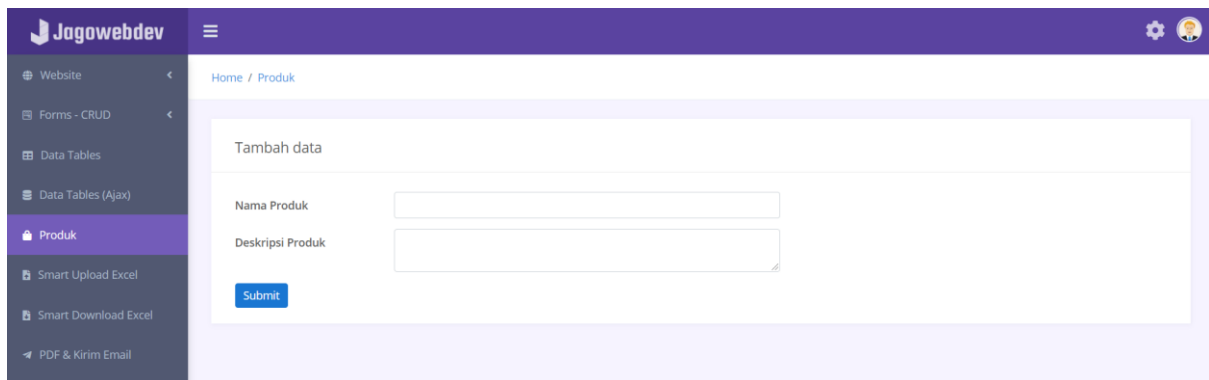
```
<?php
case 'index':
    // Script case index

case 'add':
    $data['title'] = 'Tambah data';
    load_view('views/form.php', $data);

case 'edit':
    // Script case edit
```

Perhatikan bahwa kita cukup menggunakan satu form untuk tambah dan edit data, yaitu file views/form.php. Dengan menggunakan satu form, maka maintenance akan menjadi lebih mudah.

Selanjutnya, ketika tombol + Tambah Data di klik, maka halaman akan diarahkan ke [http://localhost/admin\\_template/produk/add](http://localhost/admin_template/produk/add) dan tampilan yang dihasilkan adalah sebagai berikut:



### F.4.2. Submit Data

Selanjutnya, kita tambahkan script submit data pada case add sehingga script pada case add menjadi berikut:

```
<?php
case 'index':
    // Script index

case 'add':

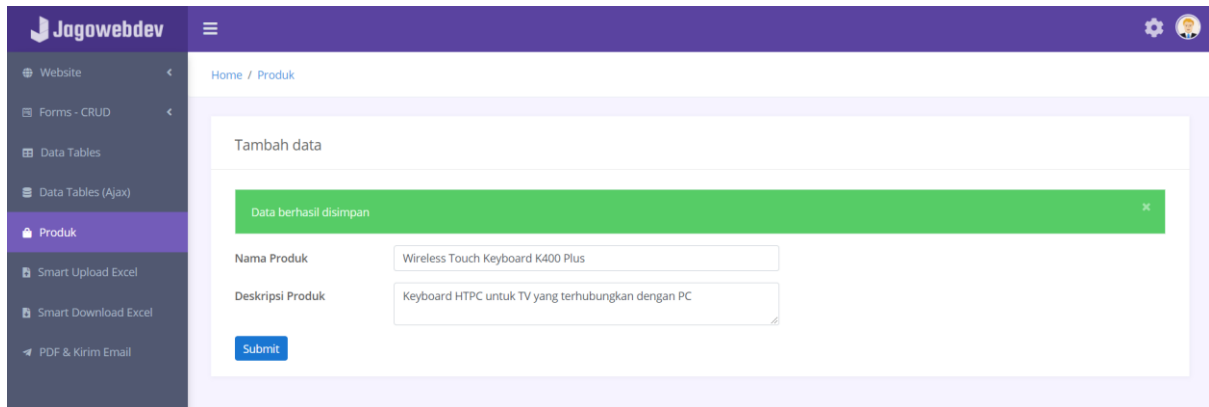
    $data['title'] = 'Tambah data';

    $message = [];
    if (!empty($_POST['submit'])) {
        $error = validate_form();
        if ($error) {
            $message['status'] = 'error';
            $message['message'] = $error;
        } else {
            $data_db['nama_produk'] = $_POST['nama_produk'];
            $data_db['deskripsi_produk'] = $_POST['deskripsi_produk'];
            $query = $db->insert('produk', $data_db);
            if ($query) {
                $message['status'] = 'ok';
                $message['message'] = 'Data berhasil disimpan';
            } else {
                $message['status'] = 'error';
                $message['message'] = 'Data gagal disimpan';
            }
        }
    }
    $data['message'] = $message;

    load_view('views/form.php', $data);

case 'edit':
    // Script edit
```

Ketika form disubmit dan data berhasil disimpan, maka akan muncul pesan bahwa data berhasil disimpan:



Skenario selanjutnya adalah, ketika data berhasil disubmit, maka posisi form adalah edit data, sehingga ketika form disubmit kembali, maka tidak menambah data, melainkan update data yang baru saja disimpan. Untuk keperluan tersebut, maka kita perlu mendapatkan data id\_produk yang baru saja tersimpan kemudian meletakkannya pada form.

Jika kita lihat kembali script views/form.php, maka letak id\_produk ada di input dengan name id di sebelah bawah tombol submit, ketika halaman tambah data dibuka, script yang muncul adalah sebagai berikut:

```
<div class="form-group row mb-0">
  <div class="col-sm-5">
    <button type="submit" name="submit" value="submit" class="btn btn-
primary">Submit</button>
    <input type="hidden" name="id" value="">
  </div>
</div>
```

Sedangkan script php pada file views/form.php adalah sebagai berikut:

```
<div class="form-group row mb-0">
  <div class="col-sm-5">
    <button type="submit" name="submit" value="submit" class="btn btn-
primary">Submit</button>
    <input type="hidden" name="id" value="<?=@$_GET['id']?>"/>
  </div>
</div>
```

Pada script form.php, agar id\_produk tidak hanya mengambil data dari variabel \$\_GET saja, kita ubah bagian @\$\_GET['id'] menjadi \$id\_produk sebagai berikut:

```
<input type="hidden" name="id" value="<?=@$id_produk?>"/>
```

Dengan model seperti itu, nantinya setelah form disubmit dan data berhasil disimpan (ditambah), input name="id" tersebut akan terisi data id\_produk dari data yang baru saja disimpan.

Selanjutnya, mari kita ubah script produk.php case add sehingga ketika data berhasil disimpan, submit data berikutnya adalah update data bukan insert data

```
$message = [];
```

---

```

$data['id_produk'] = '';
if (!empty($_POST['submit'])) {
    $error = validate_form();
    if ($error) {
        $message['status'] = 'error';
        $message['message'] = $error;
    } else {
        $data_db['nama_produk'] = $_POST['nama_produk'];
        $data_db['deskripsi_produk'] = $_POST['deskripsi_produk'];
        if (!empty($_POST['id'])) {
            $query = $db->update('produk', $data_db, ['id_produk' => $_POST['id']]);
            $data['id_produk'] = $_POST['id'];
        } else {
            $query = $db->insert('produk', $data_db);
            $data['id_produk'] = $db->lastInsertId();
        }

        if ($query) {
            $message['status'] = 'ok';
            $message['message'] = 'Data berhasil disimpan';
        } else {
            $message['status'] = 'error';
            $message['message'] = 'Data gagal disimpan';
        }
    }
}
$data['message'] = $message;

```

---

Pada script diatas, jika dibandingkan dengan script produk.php sebelumnya, terdapat tambahan script utama sebagai berikut:

---

```

if (!empty($_POST['id'])) {
    $query = $db->update('produk', $data_db, 'id_produk' => $_POST['id']);
    $data['id_produk'] = $_POST['id'];
} else {
    $query = $db->insert('produk', $data_db);
    $data['id_produk'] = $db->lastInsertId();
}

```

---

Selanjutnya bisa dicoba untuk memasukkan data kemudian Submit, kemudian, masih pada form yang sama, ubah data tersebut dan klik Submit. Cek pada halaman produk, jika berhasil maka data yang bertambah adalah data terakhir ketika kita melakukan perubahan data.

## F IV.6.5. Merapikan Script

Kita telah melakukan perubahan pada script views/form.php dimana \$\_GET['id'] kita ganti menjadi \$id\_produk, hal ini akan menyebabkan error ketika melakukan edit data, karena variabel \$id\_produk tidak ditemukan. Untuk memperbaikinya kita ubah script pada case edit, kita tambah script `$data['id_produk'] = $_GET['id'];` diatas `load_view('views/form.php', $data);` sebagai berikut:

---

```

// Script script sebelumnya
// ...

$data['title'] = 'Edit Data Produk';
$data['produk'] = $produk;

```

---

---

```
$data['message'] = $message;
$data['id_produk'] = $_GET['id'];
load_view('views/form.php', $data);
```

---

Selanjutnya, jika kita perhatikan, script pada case add dan edit isinya mirip, bedanya, pada script add perintah yang dijalankan adalah insert, sedangkan pada script edit, perintah yang dijalankan adalah update.

Agar lebih efisien dan tidak terjadi pengulangan penulisan script, kita buat sebuah fungsi simpan data yang dapat digunakan oleh case add dan edit. Fungsi tersebut kita beri nama `save_data()` dan kita letakkan di paling bawah. Adapun isi fungsi tersebut adalah sebagai berikut:

---

```
function save_data()
{
    global $db;
    $message = [];
    $id_produk = '';
    if (!empty($_POST['submit'])) {
        $error = validate_form();
        if ($error) {
            $message['status'] = 'error';
            $message['message'] = $error;
        } else {
            $data_db['nama_produk'] = $_POST['nama_produk'];
            $data_db['deskripsi_produk'] = $_POST['deskripsi_produk'];

            if (!empty($_POST['id'])) {
                $query = $db->update('produk', $data_db, ['id_produk' => $_POST['id']]);
                $id_produk = $_POST['id'];
            } else {
                $query = $db->insert('produk', $data_db);
                $id_produk = $db->lastInsertId();
            }

            if ($query) {
                $message['status'] = 'ok';
                $message['message'] = 'Data berhasil disimpan';
            } else {
                $message['status'] = 'error';
                $message['message'] = 'Data gagal disimpan';
            }
        }
    }
    return ['message' => $message, 'id_produk' => $id_produk];
}
```

---

Selanjutnya script case add kita ubah menjadi berikut:

---

```
case 'add':

    $result = ['message' => '', 'id_produk' => ''];
    if (!empty($_POST['submit'])) {
        $result = save_data();
    }

    $data['title'] = !empty($_POST['id']) || @$result['message']['status'] == 'ok'
        ? 'Edit Data Produk'
```

---

---

```
        : 'Tambah Data Produk';

$result = save_data();
$data['message'] = $result['message'];
$data['id_produk'] = $result['id_produk'];

load_view('views/form.php', $data);

case 'index':
    // Script case index
```

---

Sedangkan pada case edit kita ubah menjadi berikut:

---

```
case 'add';
    // Script pada case add

case 'edit':

    if (empty($_GET['id']))
        data_notfound();

    $result['message'] = [];
    if (!empty($_POST['submit'])) {
        $result = save_data();
    }

    $sql = 'SELECT * FROM produk WHERE id_produk = ?';
    $produk = $db->query($sql, $_GET['id'])->getJSONArray();

    if (!$produk)
        data_notfound();

    $data['title'] = 'Edit Data Produk';
    $data['produk'] = $produk;
    $data['message'] = $result['message'];
    $data['id_produk'] = $_GET['id'];
    load_view('views/form.php', $data);
```

---

Sebenarnya kita dapat mempersingkat script pada case add atau edit menjadi seperti berikut ini (contoh case add):

---

```
case 'add':

    $data['title'] = !empty($_POST['id']) ? 'Edit Data Produk' : 'Tambah Data
Produk';
    $result = save_data();
    $data['message'] = $result['message'];
    $data['id_produk'] = $result['id_produk'];

    load_view('views/form.php', $data);
```

---

Namun demikian alurnya menjadi tidak jelas karena seolah olah ketika halaman tambah data dibuka langsung menyimpan data `$result = save_data();`

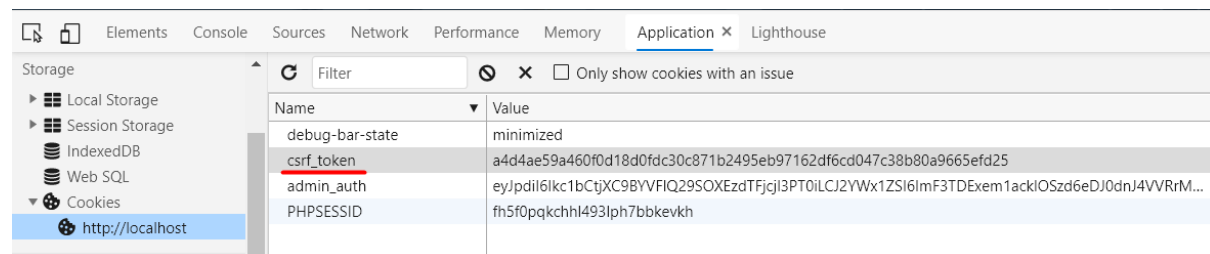
## VI. Menggunakan CSRF Token

Aplikasi Admin Template ini menyediakan built in function yang memudahkan Anda untuk menggunakan CSRF Token.

CSRF Token ini dapat disetting secara global melalui variabel `$csrf_token` yang ada di file `app/config/config.php` sebagai berikut:

```
$csrf_token = [  
    'enable_global' => true,  
    'auto_validation' => true,  
    'name' => 'csrf_token',  
    'expire' => 7200,  
    'exit_error' => false  
];
```

Untuk mengaktifkannya secara global ubah bagian `enable_global` menjadi `true`. Setelah diaktifkan, **setiap kali** halaman dibuka, aplikasi akan membuat token dan menyimpannya di cookie browser dengan nama `csrf_token`



Nama cookie ini dapat diatur melalui variabel `$csrf_token` bagian `name`

Selanjutnya, **setiap ada form dengan method post di submit**, maka akan di cek apakah token yang disimpan di cookie tadi sesuai dengan token yang disubmit, sehingga penting diperhatikan bahwa ketika fitur global ini aktif, maka setiap membuat form, kita **harus** membuat input field yang menyimpan data token. Untuk membuat field tersebut, kita cukup menggunakan fungsi `csrf_field()`.

Sebagai contoh kita memiliki form sebagai berikut:

```
<form method="post" action="/login">  
    <input type="text" name="username">  
    <input type="password" name="password">  
    <button type="submit" name="submit">Submit</button>  
    <?php csrf_field(); ?>  
</form>
```

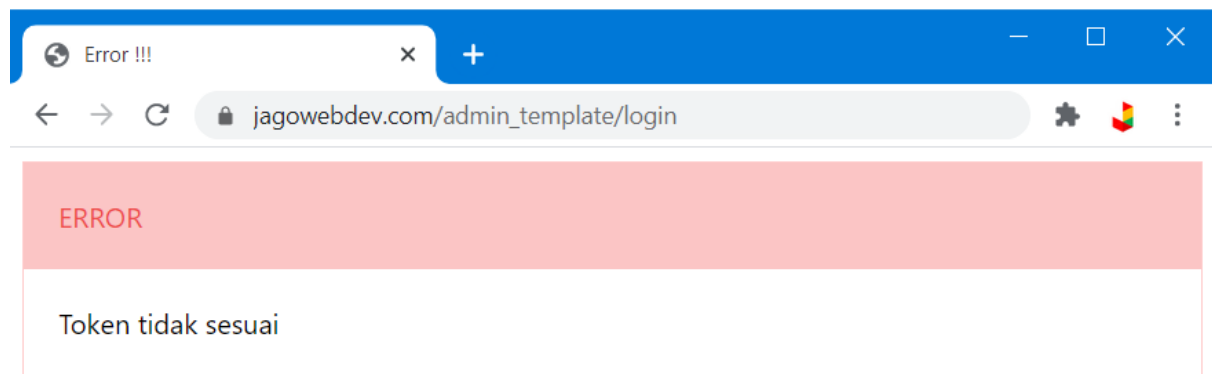
Ketika form tersebut dibuka di browser, maka kode HTML yang dihasilkan adalah sebagai berikut:

```
<form method="post" action="/login">  
    <input type="text" name="username">  
    <input type="password" name="password">  
    <button type="submit" name="submit">Submit</button>  
    <input type="hidden" name="csrf_token"  
value="a4d4ae59a460f0d18d0fdc30c871b2495eb97162df6cd047c38b80a9665efd25">  
</form>
```

Nama input yaitu `name="csrf_token"` dapat diatur melalui variabel `$csrf_token` bagian `name`



Selanjutnya, ketika form tersebut disubmit maka field `csrf_token` pada form akan dibandingkan dengan `csrf_token` pada cookie (`$_POST['csrf_token'] == $_COOKIE['csrf_token']`) jika terjadi perbedaan, maka otomatis akan keluar halaman error.



Hal ini terjadi karena nilai `exit_error` pada variabel `$csrf_token` (file `config.php`) bernilai `true`. Jika nilai `exit_error` bernilai `false`, maka sistem hanya melakukan pengecekan, selanjutnya hasil pengecekan tersebut dapat diperoleh dengan memanggil fungsi `csrf_validation()`, sebagai contoh, jika nilai `exit_error` kita buat `false`, berikut contoh pengecekan manualnya.

```
<?php
if (!empty($_POST['submit'])) {
    $cek_csrf = csrf_validation();
    if ($cek_csrf['status'] == 'error') {
        echo 'CSRF Error: ' . $cek_csrf['message'];
    }
}
?>
<form method="post" action="/login">
    <input type="text" name="username">
    <input type="password" name="password">
    <button type="submit" name="submit">Submit</button>
    <?php csrf_field(); ?>
</form>
```

## CSRF Token Manual

Selain otomatis, CSRF Token juga dapat digunakan secara manual. Caranya, pertama tama kita non aktifkan pengecekan global dengan cara mengubah nilai `enable_global` pada variabel `$csrf_token` menjadi `false`. Selanjutnya, sebelum membuat form, pertama tama kita set csrf token pada cookie, caranya, jalankan fungsi `csrf_settoken()`, selanjutnya pada form kita buat field csrf dengan menjalankan fungsi `csrf_field()` sama seperti yang kita buat pada contoh sebelumnya. Adapun contoh penerapannya adalah sebagai berikut:

```
<?php
csrf_settoken();
?>
<form method="post" action="/login">
    <input type="text" name="username">
    <input type="password" name="password">
    <button type="submit" name="submit">Submit</button>
    <?php csrf_field(); ?>
</form>
```

Selanjutnya untuk melakukan pengecekan, sama seperti pengecekan yang kita lakukan sebelumnya yaitu menggunakan fungsi `csrf_validation()` sebagai berikut:

---

```
<?php
csrf_settoken();

if (!empty($_POST['submit'])) {
    $cek_csrf = csrf_validation();
    if ($cek_csrf['status'] == 'error') {
        echo 'CSRF Error: ' . $cek_csrf['message'];
    }
}
?>
<form method="post" action="/login">
    <input type="text" name="username">
    <input type="password" name="password">
    <button type="submit" name="submit">Submit</button>
    <?php csrf_field(); ?>
</form>
```

---

**NOTE:** Contoh penerapan CSRF secara manual ada di module login (file `app/modules/login/login.php`).

## VII. Menggunakan RBAC

Aplikasi ini sudah disertakan tools RBAC untuk mempermudah Anda mengatur akses data sesuai dengan role yang dimiliki oleh user.

### VI.1. Konfigurasi

Pengaturan global tools RBAC ini ada pada file `app/config/config.php` variabel `$check_role_action`

---

```
$check_role_action = ['enable_global' => true, 'field' => 'id_user_input'];
```

---

Jika nilai `enable_global` bernilai `true`, maka ketika halaman dibuka, sistem akan otomatis mengecek apakah role user boleh melakukan aksi `add`, `edit`, atau `delete` data. Berikut metode pengecekan nya:

- **Add dan Edit.** System akan mengecek variabel `$_GET['action']`, jika variabel tersebut bernilai `add` atau `edit`, maka akan dilakukan pengecekan apakah role boleh melakukan aksi `add` dan `edit`. Sehingga penting untuk mendefinisikan url untuk aksi tambah data menggunakan url [http://localhost/nama\\_module/add](http://localhost/nama_module/add) atau [http://localhost/nama\\_module?action=add](http://localhost/nama_module?action=add) dan edit data menggunakan url [http://localhost/nama\\_module/edit](http://localhost/nama_module/edit) atau [http://localhost/nama\\_module?action=edit](http://localhost/nama_module?action=edit)
- **Delete.** System akan mengecek variabel `$_POST['delete']`, jika variabel tersebut didefinisikan maka akan dilakukan pengecekan apakah role diperbolehkan melakukan aksi `delete`, sehingga penting untuk menambahkan `<input type="hidden" name="delete">` pada form delete

**Catatan:** Pada global check (`enable_global = true`), hak akses hanya akan dicek jika role tersebut memiliki hak akses `No`, jika nilai hak akses `Own`, system tidak melakukan pengecekan, pengecekan harus dilakukan di level module.

Sebagai contoh, melanjutkan contoh sebelumnya, user biasa (username: user, pass: user) kita beri hak akses ke module produk sebagai berikut:

☒ User

Read own | Hanya Data Miliknya Sendiri ▼

Create no | TIDAK ▼

Update own | Hanya Data Miliknya Sendiri ▼

Delete own | Hanya Data Miliknya Sendiri ▼

Pada hak akses diatas, user tidak diperbolehkan untuk menambah data dan hanya dapat membaca, mengedit, dan menghapus data miliknya sendiri.

Selanjutnya, jika user membuka halaman produk dan mengklik tombol tambah data

Produk

[+ Tambah Data](#)

No	Nama Produk	Deskripsi Produk	Aksi
1	Bluetooth Multi-Device Keyboard K480	Keyboard meja wireless untuk komputer, tablet, dan smartphone	<a href="#">Edit</a> <a href="#">Delete</a>
2	USB Unifying Receiver	Receiver USB yang bisa digunakan untuk sebuah mouse atau keyboard unifying	<a href="#">Edit</a> <a href="#">Delete</a>
3	M590 Multi-Device Silent	Mouse wireless hening untuk power user	<a href="#">Edit</a> <a href="#">Delete</a>
4	Wireless Touch Keyboard K400 Plus	Keyboard HTPC untuk TV yang terhubung dengan PC	<a href="#">Edit</a> <a href="#">Delete</a>

Maka akan muncul pesan error sebagai berikut:

← → ↺ ⓘ localhost/admin\_template/produk/add 🔍 ☆ ⌵ 📁 🌐 ...

**ERROR**

Role Anda tidak diperkenankan untuk menambah data

Selanjutnya mari kita ubah hak akses user agar dapat menambahkan data

☒ User

Read own | Hanya Data Miliknya Sendiri ▼

Create yes | YA ▼

Update own | Hanya Data Miliknya Sendiri ▼

Delete own | Hanya Data Miliknya Sendiri ▼

Dengan hak akses diatas, user boleh menambah data, namun tetap hanya dapat membaca, mengedit, dan menghapus data miliknya sendiri.

## VI.2. Read Data

Untuk hak akses read data, agar nantinya dapat dilakukan pengecekan otomatis, kita perlu tambahkan kolom `id_user_input` pada tabel database, selanjutnya setiap kali menyimpan data, kita simpan data `id_user` ke kolom tersebut. Sebagai contoh kita ubah script pada module produk bagian case 'add' sehingga dapat menyimpan `id_user` user yang menginput data, script sebelumnya adalah sebagai berikut:

---

```
global $db;
$message = [];
$id_produk = '';
if (!empty($_POST['submit'])) {
    $error = validate_form();
    if ($error) {
        $message['status'] = 'error';
        $message['message'] = $error;
    } else {
        $data_db['nama_produk'] = $_POST['nama_produk'];
        $data_db['deskripsi_produk'] = $_POST['deskripsi_produk'];
        if (!empty($_POST['id'])) {
            $query = $db->update('produk', $data_db, ['id_produk' =>
$_POST['id']]);
            $id_produk = $_POST['id'];
        } else {
            $query = $db->insert('produk', $data_db);
            $id_produk = $db->lastInsertId();
        }

        if ($query) {
            $message['status'] = 'ok';
            $message['message'] = 'Data berhasil disimpan';
        } else {
            $message['status'] = 'error';
            $message['message'] = 'Data gagal disimpan';
        }
    }
}
```

---

Selanjutnya kita tambahkan script untuk menyimpan `id_user` yang menginput, sehingga script simpan data menjadi sebagai berikut:

---

```
// Script lainnya
if (!empty($_POST['id'])) {
    $data_db['id_user_update'] = $_SESSION['user']['id_user'];
    $query = $db->update('produk', $data_db, ['id_produk' => $_POST['id']]);
    $id_produk = $_POST['id'];
} else {
    $data_db['id_user_input'] = $_SESSION['user']['id_user'];
    $query = $db->insert('produk', $data_db);
    $id_produk = $db->lastInsertId();
}
// Script lainnya
```

---

Selanjutnya, login sebagai user dan coba untuk menyimpan data berikut:

Tambah Data Produk

Nama Produk

K380 Multi-Device Bluetooth Keyboard

Deskripsi Produk

Keyboard minimalis untuk komputer, tablet, dan ponsel

Submit

Setelah klik submit, maka tabel produk pada database akan tampak sebagai berikut:

id_produk	nama_produk	deskripsi_produk	id_user_input	tgl_input	id_user_edit	tgl_edit
1	Bluetooth Multi-Device Keyboard K480	Keyboard meja wireless untuk komputer, tablet,...	(NULL)	2021-01-29	(NULL)	2021-01-29
2	USB Unifying Receiver	Receiver USB yang bisa digunakan untuk sebuah...	(NULL)	2021-01-29	(NULL)	2021-01-29
3	M590 Multi-Device Silent	Mouse wireless hening untuk power user	(NULL)	2021-01-29	(NULL)	2021-01-29
4	Wireless Touch Keyboard K400 Plus	Keyboard HTPC untuk TV yang terhubungkan de...	(NULL)	2021-01-29	(NULL)	2021-01-29
5	K380 Multi-Device Bluetooth Keyboard	Keyboard minimalis untuk komputer, tablet, dan...	2	2021-01-29	(NULL)	2021-01-29

Pada contoh diatas, pada id\_produk 5 yang baru saja kita tambahkan, id\_user\_input nya bernilai 2, yaitu id\_user dari user yang login. Untuk baris yang lain (yang bernilai NULL) kita biarkan apa adanya.

Sebelumnya hak akses user untuk read data adalah own, namun ketika membuka halaman produk, semua data muncul

Produk

+ Tambah Data

No	Nama Produk	Deskripsi Produk	Aksi
1	Bluetooth Multi-Device Keyboard K480	Keyboard meja wireless untuk komputer, tablet, dan smartphone	<a href="#">Edit</a> <a href="#">Delete</a>
2	USB Unifying Receiver	Receiver USB yang bisa digunakan untuk sebuah mouse atau keyboard unifying	<a href="#">Edit</a> <a href="#">Delete</a>
3	M590 Multi-Device Silent	Mouse wireless hening untuk power user	<a href="#">Edit</a> <a href="#">Delete</a>
4	Wireless Touch Keyboard K400 Plus	Keyboard HTPC untuk TV yang terhubungkan dengan PC	<a href="#">Edit</a> <a href="#">Delete</a>
5	K380 Multi-Device Bluetooth Keyboard	Keyboard minimalis untuk komputer, tablet, dan ponsel	<a href="#">Edit</a> <a href="#">Delete</a>

Hal ini terjadi karena kita belum melakukan filter pada data. Untuk melakukannya kita tambahkan filter pada query pengambilan data. Pada case index, terdapat query sebagai berikut:

```
$sql = 'SELECT * FROM produk';
```

Selanjutnya kita tambahkan fungsi where\_own() sehingga bentuk query menjadi berikut:

---

```
$sql = 'SELECT * FROM produk' . where_own();
```

---

Fungsi where\_own() ini ada di file system/libraries/rbac.php fungsi ini akan otomatis menambahkan klausa WHERE sebagai berikut:

---

```
WHERE id_user_input = . $_SESSION['user']['id_user'];
```

---

Sehingga jika dijalankan, query diatas menjadi:

---

```
$sql = 'SELECT * FROM produk WHERE id_user_input = 2';
```

---

Perhatikan bahwa pada query diatas yang dilakukan pengecekan adalah kolom id\_user\_input. Fungsi where\_own() menggunakan kolom ini karena pada konfigurasi `$check_role_action` bagian `field` bernilai id\_user\_input. Selanjutnya, ketika kita jalankan, maka yang muncul hanya data dengan id\_user\_input yang bernilai 2 sebagai berikut

Home / Produk

Produk

[+ Tambah Data](#)

No	Nama Produk	Deskripsi Produk	Aksi
1	K380 Multi-Device Bluetooth Keyboard	Keyboard minimalis untuk komputer, tablet, dan ponsel	<a href="#">Edit</a> <a href="#">Delete</a>

## VI.3. Edit Data

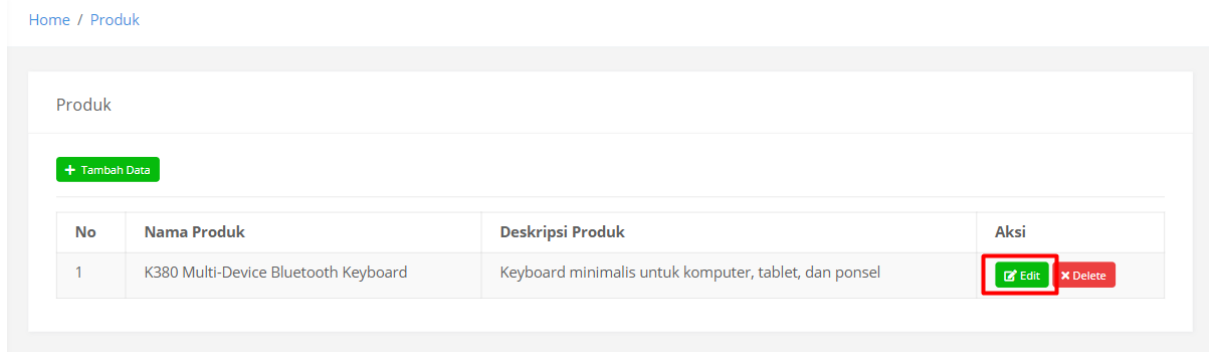
Pada setting role user yang telah kita definisikan sebelumnya, hak akses user untuk update\_data adalah Own yang artinya hanya bisa mengupdate data miliknya sendiri. Agar user hanya dapat mengedit data miliknya sendiri, kita tambahkan fungsi cek\_hakakses('update\_data') pada case 'edit' sebagai berikut:

---

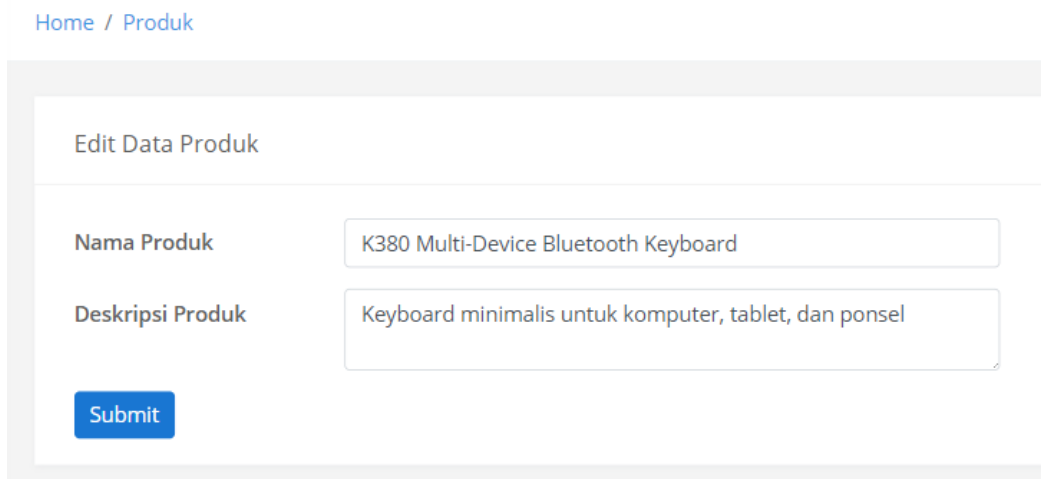
```
case 'add' :  
    // Script add  
  
case 'edit':  
    cek_hakakses('update_data');  
  
    if (empty($_GET['id']))  
        data_notfound();  
  
    $result['message'] = [];  
  
    // Script lainnya
```

---

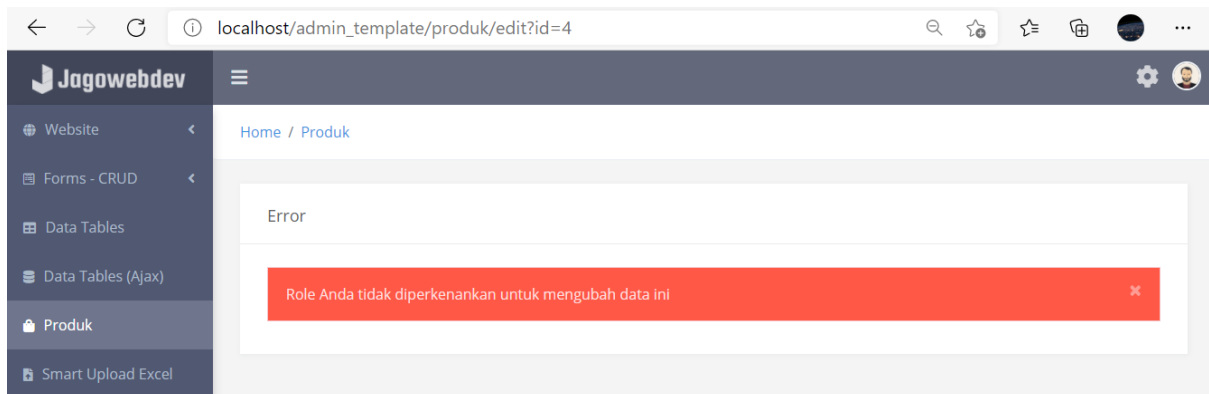
Selanjutnya jika kita klik tombol edit:



Maka kita akan diarahkan ke url [http://localhost/admin\\_template/produk/edit?id=5](http://localhost/admin_template/produk/edit?id=5) dengan tampilan sebagai berikut:



Jika parameter id kita ubah menjadi misal 4, sehingga url menjadi [http://localhost/admin\\_template/produk/edit?id=4](http://localhost/admin_template/produk/edit?id=4) maka yang kita dapati adalah sebagai berikut:



Fungsi `cek_hakakses()` ini berada pada file `system/libraries/csrf.php` ketika dijalankan (pada contoh diatas) fungsi akan mengecek apakah produk dengan `id_produk` 5 adalah milik user yang sedang login (`id_user` 2) sehingga boleh diedit, yaitu dengan melakukan pengecekan kolom `id_user_input`.

Fungsi `cek_hakakses()` tahu yang dicek tabel produk karena kita tidak mendefinisikan nama tabel pada fungsi tersebut, sehingga fungsi tersebut berasumsi bahwa nama tabel sama dengan nama module yaitu produk. Selanjutnya pada tabel produk, fungsi akan mengambil data yang kolom `id_produk` nya bernilai 5. Fungsi tahu kolom yang digunakan adalah `id_produk` karena kita tidak mendefinisikan nama kolom, sehingga fungsi berasumsi bahwa kolom id untuk data produk adalah `id_ + nama_tabel` yaitu `id_produk`.

Parameter lengkap untuk fungsi hak akses adalah:

```
cek_hakakses(aksi, nama_tabel|nama_kolom, nama_kolom_yang_dicek);
```

Sehingga, pada contoh diatas, jika fungsi cek\_hakakses() kita tulis lengkap, bentuknya menjadi

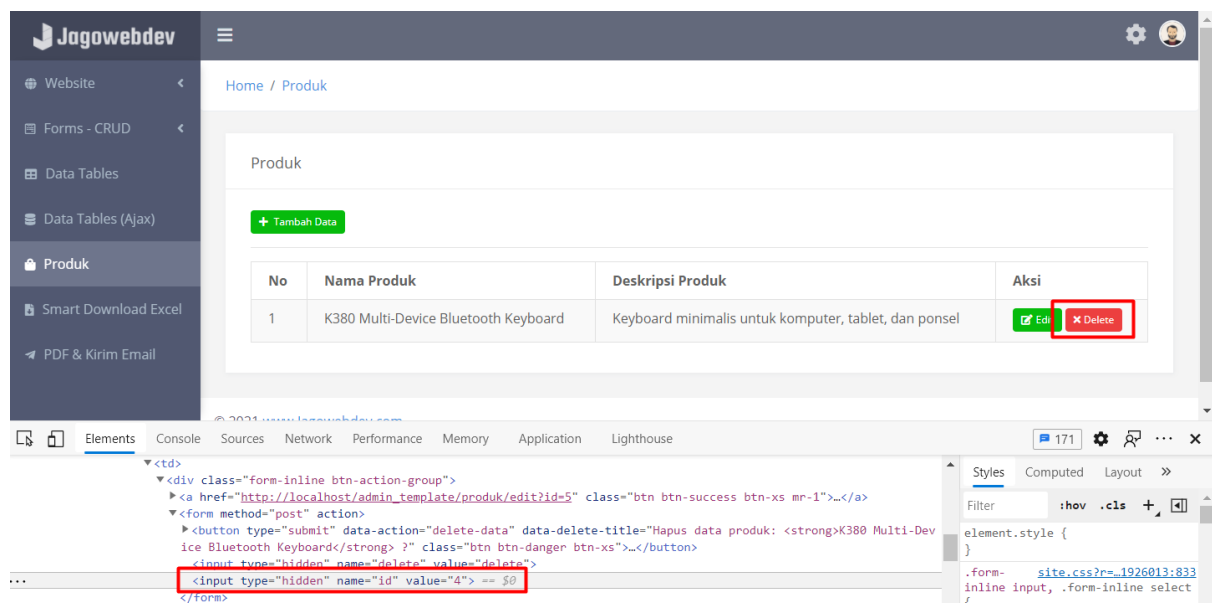
```
cek_hakakses('update_data', 'produk|id_produk', 'id_user_input');
```

## VI.4. Delete Data

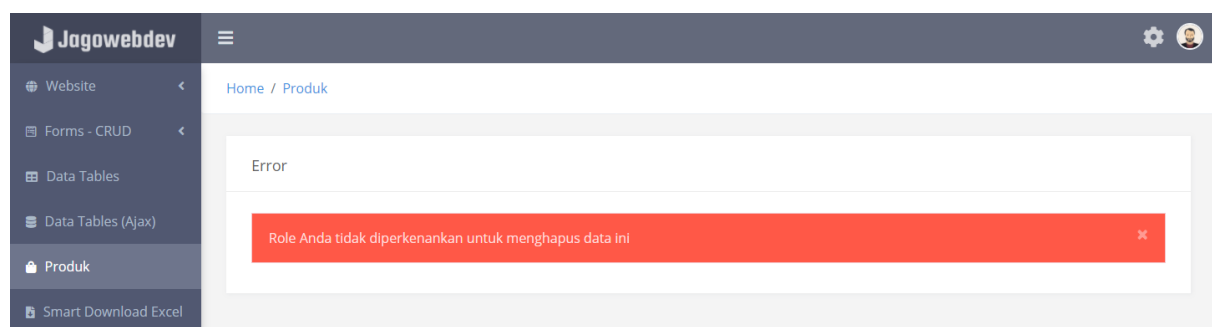
Selanjutnya mari kita atur parameter untuk delete data agar sesuai dengan hak akses yang telah ditetapkan. Pada contoh diatas, hak akses delete data untuk role user adalah own yang artinya user hanya dapat menghapus data miliknya sendiri. Agar dapat memenuhi kriteria tersebut, caranya sama seperti edit, yaitu kita cukup tambahkan fungsi cek\_hakakses('delete\_data') sebagai berikut:

```
case 'index':  
    $message = [];  
    if (!empty($_POST['delete'])) {  
        cek_hakakses('delete_data');  
        // Script lainnya
```

Selanjutnya jika kita tes dengan mengubah nilai id dan klik delete



Maka hasil yang kita peroleh adalah sebagai berikut:



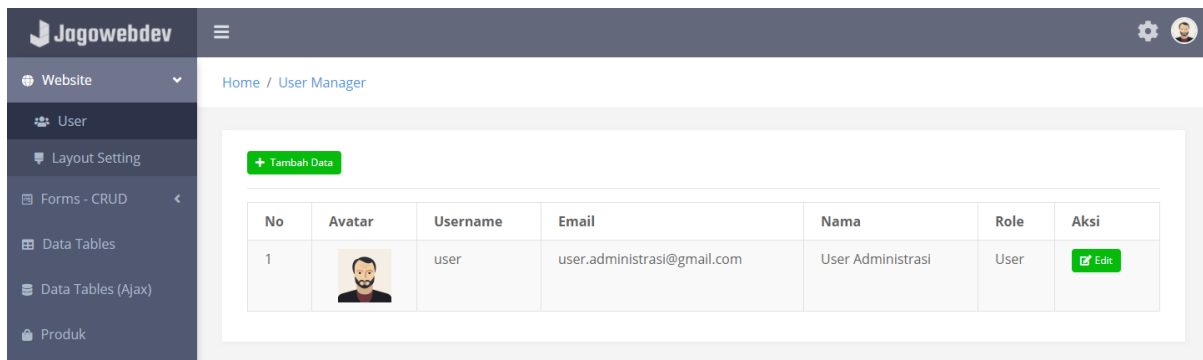
## VI.5. Studi Kasus




Contoh lain penerapan RABC adalah pada module user (app/modules/user.php).

## Add Data

Pada module ini, user hanya dapat melihat data miliknya sendiri. Misal ketika user (username: user, pass: user) login dan membuka menu user, maka akan tampil data miliknya sendiri sebagai berikut:



No	Avatar	Username	Email	Nama	Role	Aksi
1		user	user.administrasi@gmail.com	User Administrasi	User	<a href="#">Edit</a>

Untuk memperoleh hasil seperti itu, kita tambahkan fungsi `where_own('id_user')` pada query pengambilan data user sebagai berikut:

```
$sql = 'SELECT * FROM user LEFT JOIN role USING(id_role)' . where_own('id_user');
```

Perhatikan bahwa pada query diatas, kita tambahkan argumen `id_user` pada fungsi `where_own()`, dengan argumen ini, maka fungsi akan mengecek kolom `id_user` untuk dicocokkan dengan `id_user` yang login. Hal ini berbeda dengan contoh pada module produk dimana kita cukup menuliskan fungsi `where_own()` tanpa argumen, karena pada module produk, kolom yang dicek adalah `id_user_input` (sesuai dengan parameter field pada variabel `$check_role_action` yang ada di file `app/config/config.php`)

## Edit Data

Agar edit data user sesuai dengan role yang telah ditetapkan, pada case edit, kita tambahkan fungsi

```
cek_hakakses('update_data', 'user', 'id_user');
```

Perhatikan bahwa fungsi `cek_hakakses()` diatas kita tambahkan parameter kolom `id_user` karena kolom yang ingin kita cek berbeda dengan kolom default yaitu `id_user_input`. Sedangkan parameter tabel yaitu `user` boleh ditambahkan boleh tidak, karena nama tabel sudah sama dengan nama module yaitu `user` dan nama kolom untuk pengambilan data juga sudah sesuai yaitu `id_ + nama tabel` atau `id_user`, jika tidak ingin menambahkan parameter tabel, maka bentuk fungsi menjadi:

```
cek_hakakses('update_data', null, 'id_user');
```

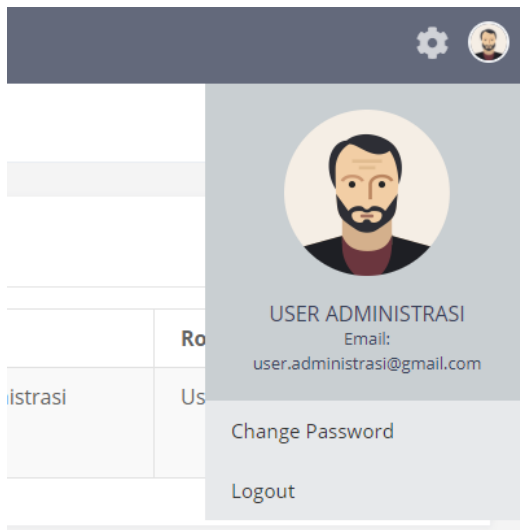
## Delete Data

Untuk hak akses delete data, sama dengan aksi edit data, yaitu kita tambahkan fungsi `cek_hakakses` dengan parameter lengkap, sebagai berikut:

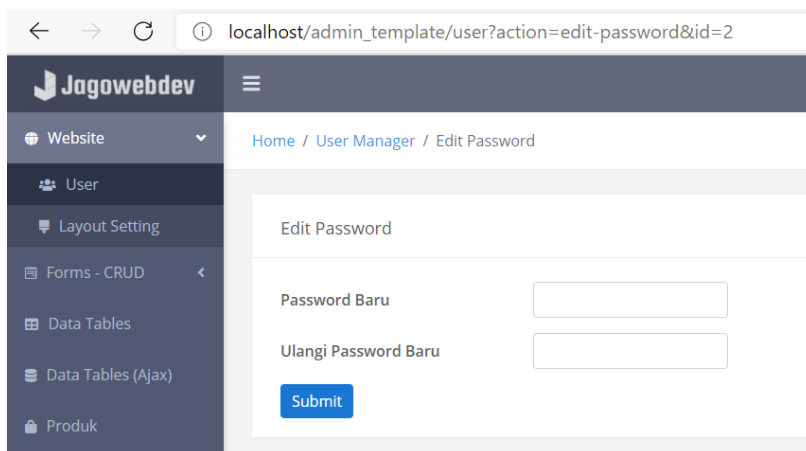
```
if (!empty($_POST['delete']))
{
    cek_hakakses('delete_data', 'user', 'id_user');
```

## Update Password

Selanjutnya, pada menu user, update passwordnya terpisah dari menu edit dan delete, yaitu ada di menu user di pojok kanan atas



Ketika menu Change Password di klik, maka kita akan diarahkan ke url [http://localhost/admin\\_template/user?action=edit-password&id=2](http://localhost/admin_template/user?action=edit-password&id=2) dengan tampilan halaman sebagai berikut:



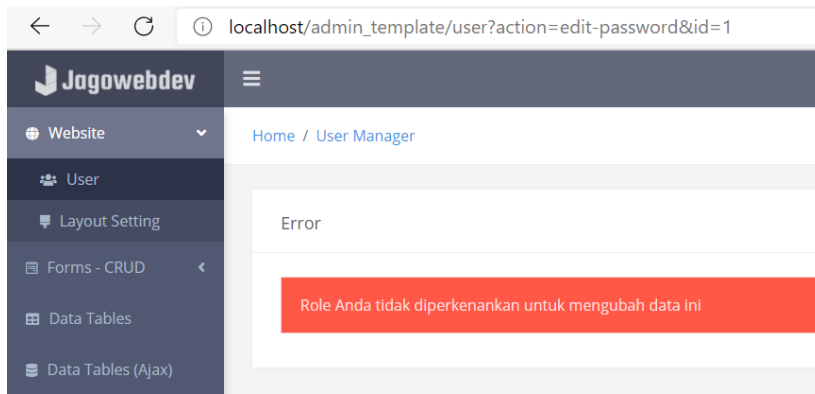
Pada kondisi diatas, user dapat mengubah id pada url sehingga dapat mengubah password user lain, untuk mengatasi hal tersebut, sama seperti pada edit data yaitu ditambahkan fungsi `cek_hakakses()` sebagai berikut:

---

```
cek_hakakses('update_data', null, 'id_user');
```

---

Sehingga jika user mencoba mengakses data user lain misal dengan mengganti id pada url menjadi 1 maka akan muncul error sebagai berikut:



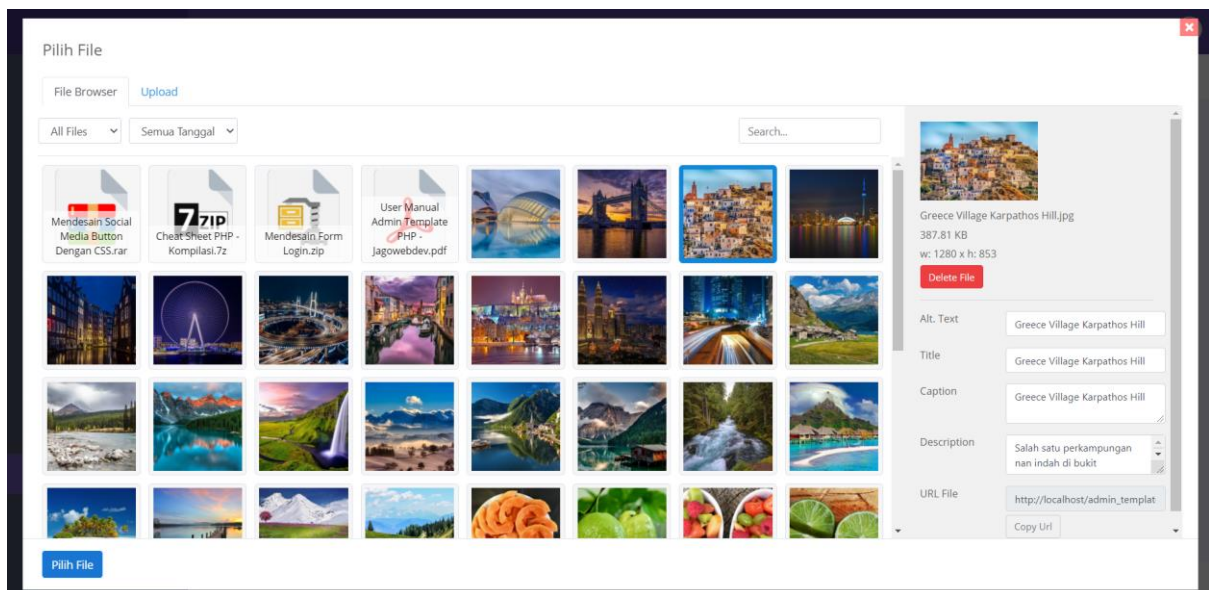
## Contoh Lain

Contoh lain penerapan RBAC ini dapat dilihat pada module datatables dan datatables-ajax

## VIII. File Picker

Kami telah menyertakan plugin File Picker untuk memudahkan Anda memilih file untuk berbagai keperluan. File Picker ini kami kembangkan sendiri sehingga kompatibel dengan aplikasi Admin Template ini, disamping juga dapat digunakan di luar admin template ini.

File Picker ini berbeda dengan File Manager. Jika File Manager lebih seperti file browser, File Picker ini menggunakan database untuk menyimpan data file beserta atributnya. Tabel database yang digunakan adalah tabel **file\_picker** Adapun tampilannya adalah sebagai berikut:



### 1. Lokasi Plugin

Plugin ini berlokasi di public/vendors/jwdfilepicker

### 2. Konfigurasi

Untuk dapat menggunakan plugin ini, perlu melakukan beberapa konfigurasi yaitu:

- File app/config/config.php

```
$config = [
```

---

```

/* File Picker */
, 'thumbnail' => [
    'small' => ['w' => 250, 'h' => 250],
    'medium' => ['w' => 450, 'h' => 450]
],
'filepicker_upload_path' => realpath(__DIR__ . '/../..') . '/public/files/uploads/'
, 'filepicker_upload_url' => 'http://localhost/admin_template/public/files/uploads/'
, 'filepicker_server_url' => 'http://localhost/admin_template/filepicker/'
, 'filepicker_icon_url' => 'http://localhost/admin_template/public/images/filepicker_images/'
];

```

---

- thumbnail: mengatur ukuran thumbnail ketika mengupload file
- filepicker\_upload\_path: lokasi folder dimana file akan diupload
- filepicker\_upload\_url: lokasi url folder dimana file diupload
- filepicker\_icon\_url: url lokasi file icon yang digunakan oleh plugin ini
- item\_per\_page: menampilkan jumlah item ketika halaman di scroll.

b. File app/themes/modern/header.php

---

```

<script type="text/javascript">
    ...
    var filepicker_server_url = "<?=$config['filepicker_server_url']?>";
    var filepicker_icon_url = "<?=$config['filepicker_icon_url']?>";
</script>

```

---

Pendefinisian ini bertujuan agar variabel `filepicker_server_url` dan `filepicker_icon_url` di javascript dapat digunakan secara global, salahsatunya di file `jwdfilepicker-default.js`

c. File public/themes/modern/js/jwdfilepicker-default.js

---

```

$(document).ready(function() {
    jwdfilepicker.setDefaults({
        server_url : filepicker_server_url,
        icon_url : filepicker_icon_url
    });
});

```

---

Pendefinisian ini bertujuan untuk mendefinisikan secara global parameter `server_url` dan `icon_url` yang diperlukan oleh plugin. Selain global, pendefinisian ini juga dapat dilakukan secara manual ketika plugin dieksekusi.

d. Scroll

Untuk menampilkan daftar file, plugin ini tidak menggunakan pagination, tetapi infinity scroll, jumlah item yang ditampilkan ketika awal plugin dibuka maupun ketika halaman dicroll di tentukan oleh konfigurasi `item_per_page` yang ada di `app/config/config.php`

### 3. Eksekusi Plugin

Contoh fungsi untuk mengeksekusi plugin adalah sebagai berikut:

---

```

jwdfilepicker.init({
    title : 'Pilih File',
    filter_file : '',

```

---

---

```
onSelect: function ($elm) {  
    var meta = JSON.parse($elm.find('.meta-file').html());  
    $('.filename').val(meta.nama_file);  
    $('.id-file-picker').val(meta.id_file_picker);  
}  
});
```

---

Parameter `filter_file` digunakan untuk memfilter file yang akan ditampilkan pada file browser. Opsi yang tersedia adalah:

- `image`: hanya menampilkan file image
- `video`: hanya menampilkan file video
- `audio`: hanya menampilkan file audio
- `archive`: hanya menampilkan file archive (.zip, .rar, .7z, dan .gz)
- `document`: hanya menampilkan dokumen (pdf dan office / open office)

Contoh penggunaan filter: `filter_file : 'image video'` atau `filter_file : 'image'`

Contoh eksekusi plugin dapat dilihat di module File Picker Manager (app/modules/filepicker), Gallery Drag n Drop (app/modules/gallery), Artikel (app/modules/artikel), dan Stream Download (app/modules/filedownload).

#### 4. Tipe File

File Picker ini mendukung semua tipe file, baik image, video, audio, dokumen, dll. Jika berbentuk image, maka akan ditampilkan preview image tersebut, jika bukan image, maka akan ditampilkan thumbnail icon yang sesuai beserta nama file nya. Thumbnail icon ini ada di folder `public/images/filepicker_images`.

#### 5. Upload File

Plugin File Picker ini sudah disertakan drag and drop file uploader. File upload ini dapat digunakan untuk mengupload satu atau banyak file sekaligus. Jika yang diupload adalah file image, maka system akan otomatis membuat dua thumbnail untuk image tersebut yaitu small dan medium, dengan demikian akan diperoleh tiga file, yaitu file asli, file asli\_small, dan file asli\_medium. Ukuran thumbnail ini secara default 250px untuk thumbnail small, dan 450px untuk thumbnail medium (disesuaikan dengan nilai yang ada di variabel `$config`). Jika yang diupload bukan image, maka file akan diupload apa adanya.

#### 6. Dependency

Plugin ini memerlukan beberapa dependency, yaitu:

a. jQuery

`public/vendors/jquery`

b. Dropzone

`public/vendors/dropzone`

c. Sweet Alert

`public/vendors/sweetalert`

- d. PHP Image Workshop yang digunakan untuk membuat thumbnail image

app/libraries/vendors/imageworkshop

## Mengelola File

Anda dapat mengelola file yang telah diupload melalui menu File Picker > File Picker Manager. Di halaman ini, Anda dapat upload, edit, maupun delete file yang telah diupload.

## Tiny MCE

Plugin File Picker ini juga dapat diintegrasikan pada Tiny MCE, WYSIWYG text editor paling populer. Contoh penggunaannya ada di module artikel (menu File Picker > Artikel). Untuk konfigurasinya perlu menyertakan file public/themes/modern/js/artikel.js dan file public/themes/modern/js/filepicker-tinymce.js

# IX. Menggunakan Email

Aplikasi ini sudah menyertakan library PHP Mailer untuk mengirim email. Konfigurasi email yang ada di aplikasi ini dapat digunakan untuk mengirim email menggunakan berbagai layanan, mulai dari email Shared Hosting, Email Hosting, GMAIL standard authentication (less secure application), Gmail OAuth 2, dan AmazonSES

Untuk konfigurasi OAuth 2 token pada GMAIL dapat dibaca di file PDF Setting OAuth2 Gmail.pdf yang disertakan pada user manual. Konfigurasi email ada di file app/config/email.php. Untuk contoh scriptnya bisa melihat script kirim email ada di menu PDF & Send Email (app/modules/pdfkirimemail.php).

# X. Library

Admin Template Jagowebdev ini sudah disertakan beberapa library php yang siap digunakan diantaranya:

1. PHP Mailer,
2. PHPXLSX Writer,
3. Spout PHP Excel Reader dan Writer
4. MPDF

Library tersebut disimpan di folder app/libraries/vendors

Selain library php, Admin Template ini juga sudah menyertakan library javascript dan css, diantaranya adalah datatables, bootstrap, bootbox, sweetalert2, dll. Library tersebut disimpan di folder public/vendors.

# XI. Penutup

Semoga panduan ini dapat membawa manfaat bagi sobat semuanya dan semoga kedepannya lebih dapat disempurnakan lagi. Jika ada kritik dan saran mohon untuk dapat menghubungi kami melalui:

- Email: [support@jagowebdev.com](mailto:support@jagowebdev.com)
- Email: [prawoto.hadi@gmail.com](mailto:prawoto.hadi@gmail.com)
- Whats App: 0856 136 3962

Terima kasih,  
27 Mei 2021

Salam  
Agus Prawoto Hadi