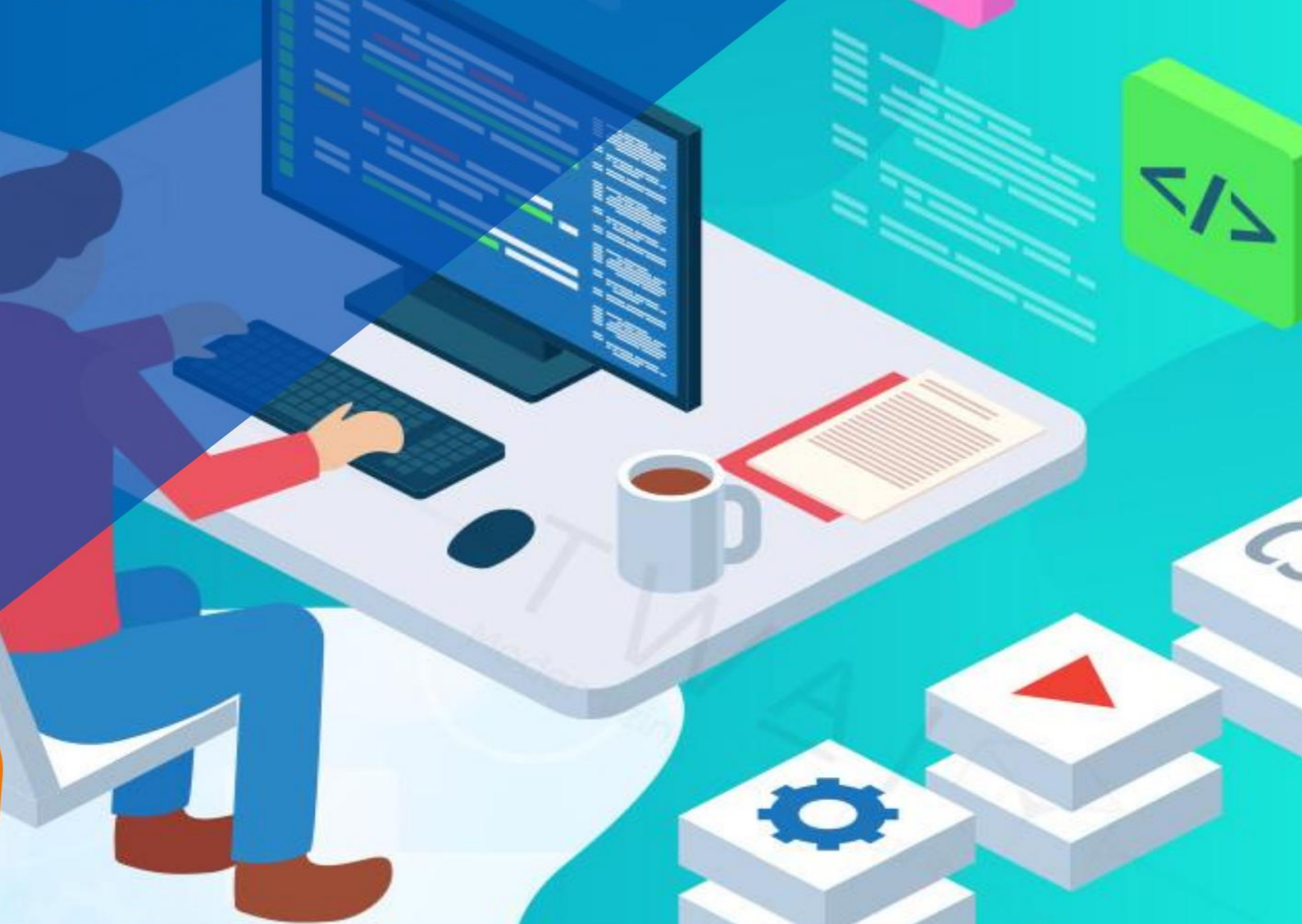


PEMROGRAMAN WEB DAN PERANGKAT BERGERAK

XII
REKAYASA PERANGKAT LUNAK
SEMESTER GANJIL



SMK NEGERI 2 TRENGGALEK

- MVC (*MODEL VIEW CONTROLLER*) PADA PHP -

- CRUD (*CREATE READ UPDATE DELETE*) -

A. KOMPETENSI DASAR

3.23 Menerapkan aplikasi web dengan model view controller (mvc)

4.23 Membuat aplikasi web menggunakan model view controller (mvc)

B. TUJUAN

Setelah proses belajar, berdiskusi, dan menggali informasi, peserta didik diharapkan mampu :

1. Menerapkan konsep MVC pada pembuatan web dinamis (PHP)
2. Membuat aplikasi web dinamis (PHP) dengan menggunakan MVC.
3. Membuat aplikasi web dengan menggunakan prinsip CRUD dengan menggunakan konsep MVC

C. ALOKSI WAKTU

13 x 45 menit

D. DASAR TEORI

1. MEMBANGUN WEB BERGAYA MVC

a. Model dengan *database*

Untuk studi kasus dengan *database*, kita akan menggunakan DBMS MySQL, silahkan buka phpmyadmin dan buat *database* dengan nama **phpmvc** dan beri nama tabel-nya **data_siswa**.

1) Tambahkan menu Data Siswa pada *website* yang telah kalian buat.

a) Buat Controller **Siswa.php** di folder **controller**.

app > controller > Siswa.php > ...

```
1  <?php
2  class Siswa extends Controller
3  {
4      public function index()
5      {
6          $data['judul'] = "Siswa";
7          $data['siswa'] = $this->model("Siswa_model")->getAllBlog();
8          $this->view('templates/header', $data);
9          $this->view('siswa/index', $data);
10         $this->view('templates/footer');
11     }
12 }
```

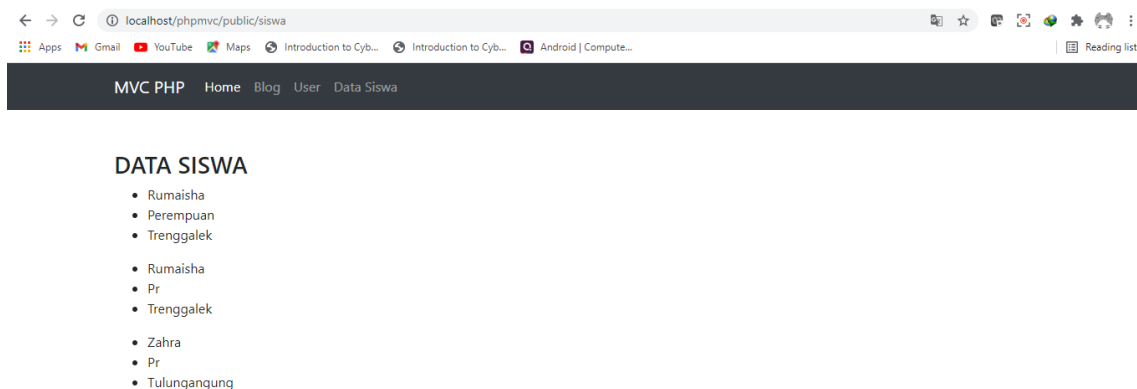
Cara yang kita gunakan tidak jauh berbeda dengan pengambilan data dari *model* sebelumnya, bisa dilihat kita akan mengambil data dari class **Blog_model** dengan method **getAllBlog()**.

b) Buat Model **Siswa_model.php** di folder **models** sama dengan **Blog_model.php**.

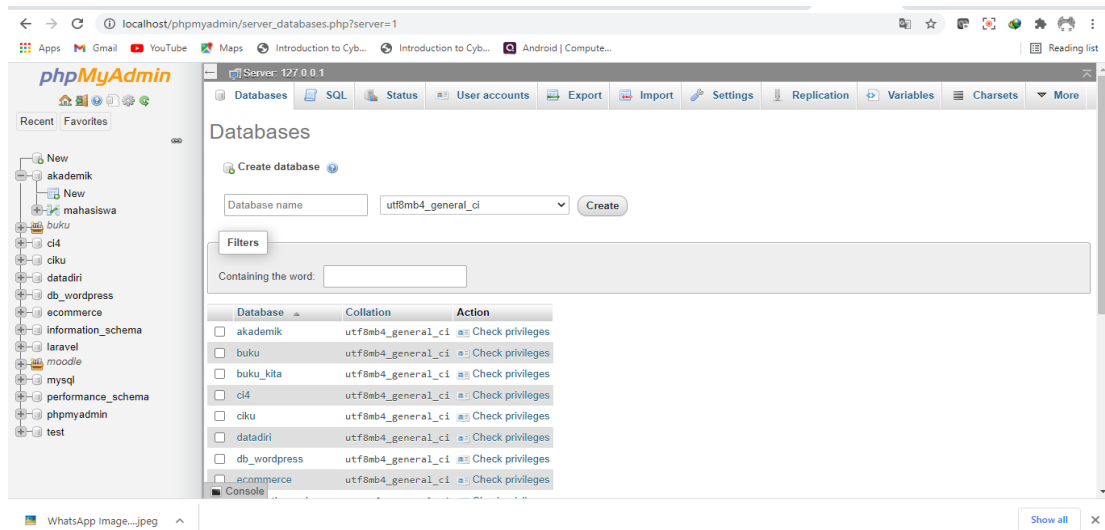
c) Buat view baru **siswa/index.php** di folder **view**.

```
12 <body>
13     <div class="container mt-5">
14         <div class="row">
15             <div class="col-6">
16                 <h3>DATA SISWA</h3>
17                 <?php foreach ($data['siswa'] as $siswa) : ?>
18                     <ul>
19                         <li><?=$siswa['nama']; ?></li>
20                         <li><?=$siswa['jenis_kelamin']; ?></li>
21                         <li><?=$siswa['alamat']; ?></li>
22                     </ul>
23                 <?php endforeach; ?>
24             </div>
25         </div>
26     </div>
27 </body>
```

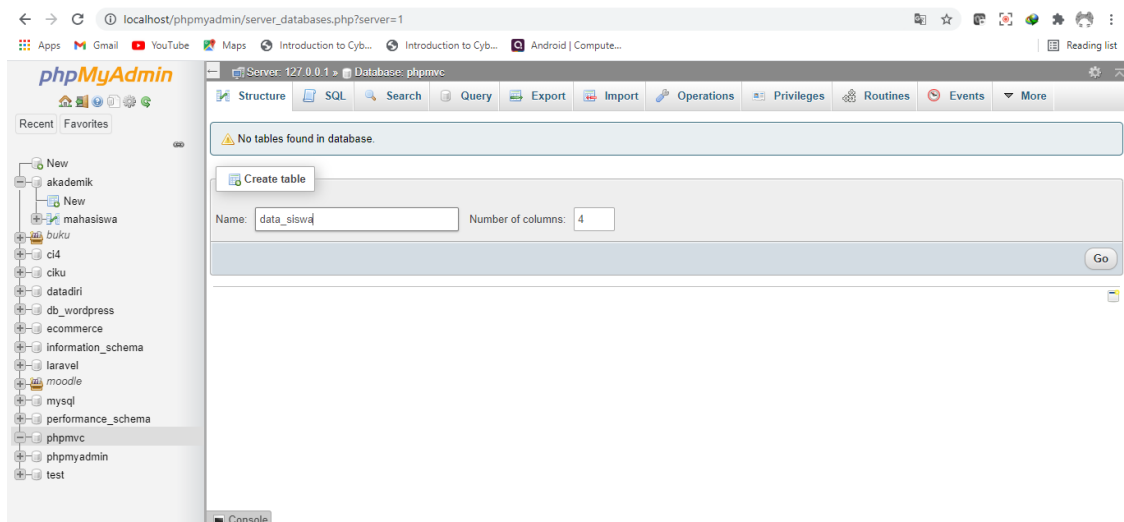
Tampilan menu Data Siswa mencontoh tampilan pada menu Blog



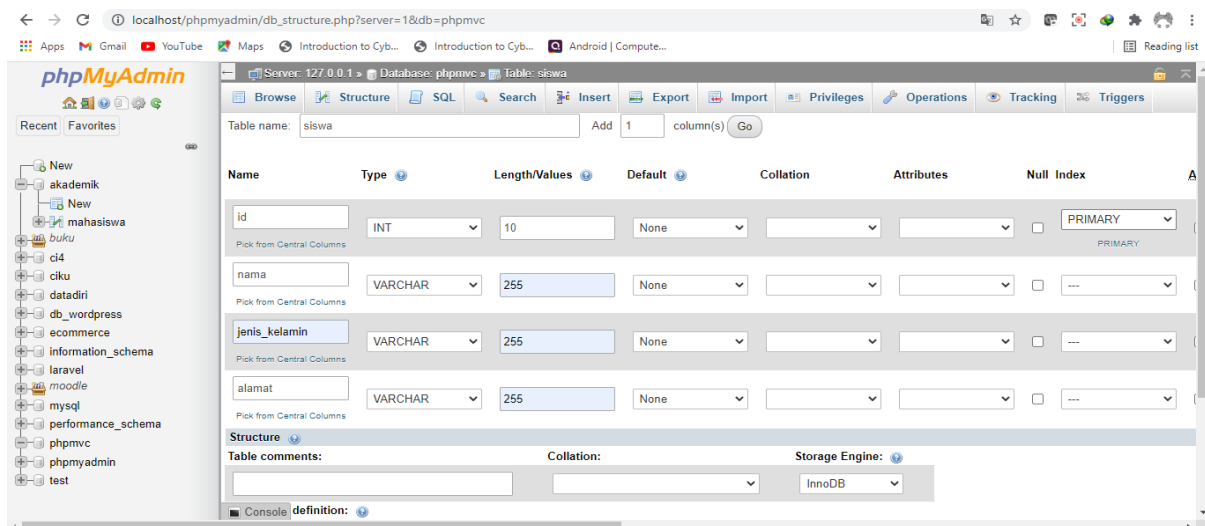
2) Buka localhost/phpMyAdmin klik new dan buat *database* baru dan beri nama **phpmvc**



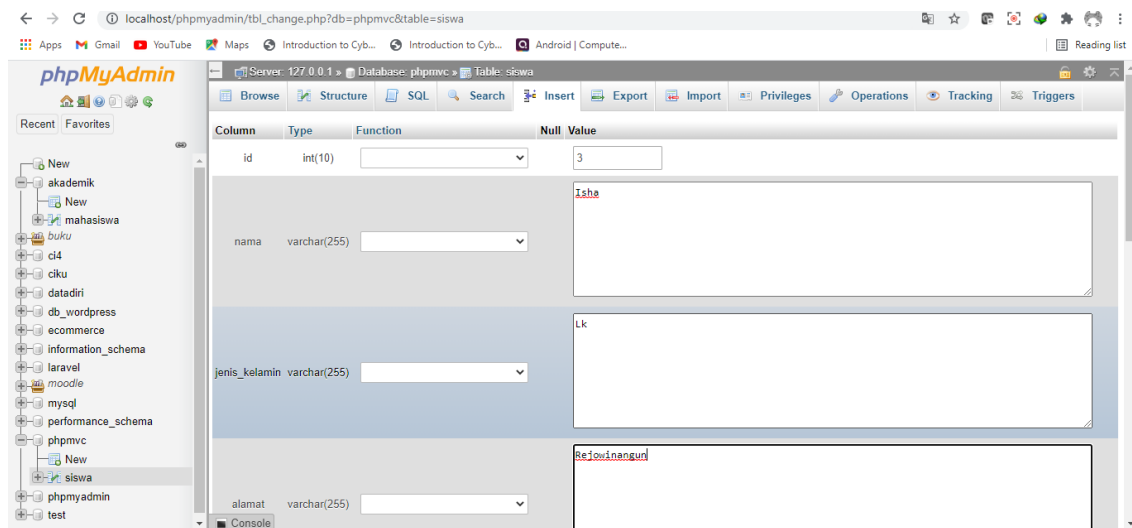
3) Buat tabel dengan kolom dan beri nama **data_siswa**



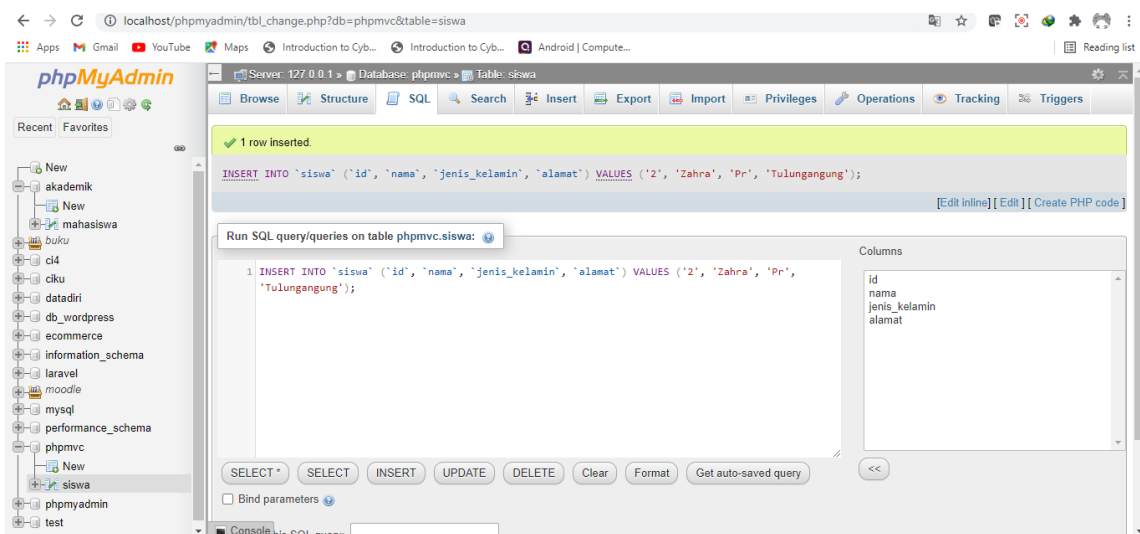
4) Lengkapi *properties* struktur dari setiap query seperti berikut



5) Kemudian tambahkan secara manual data kedalam db yang dibuat



kalau udah seperti dibawah ini kita udah punya data yang siap untuk diambil oleh model kita



Koneksi database dengan PDO (PHP DATABASE OBJECT)

Jadi, sebenarnya dengan menggunakan PDO kita bisa lebih mudah dalam megutak-atik data dalam *database*, sebenarnya menggunakan mysqli pun bisa tapi PDO lebih direkomendasikan karena beberapa orang bilang PDO bisa universal tidak harus MySQL, dia bisa juga dipake di *database* lain seperti PostgreSQL atau SQLite. Juga katanya PDO ini lebih cepat, lebih solid dan juga aman.

1) Edit Siswa_model.php

app > models > Siswa_model.php > ...

```
1  <?php
2  class Siswa_model
3  {
4      private $dbh;
5      private $stmt;
6      public function __construct()
7      {
8          //data source name
9          $dsn = "mysql:host=127.0.0.1;dbname=phpmvc";
10         try {
11             $this->dbh = new PDO($dsn, 'root', '');
12         } catch (PDOException $e) {
13             die($e->getMessage());
14         }
15     }
16     public function getAllBlog()
17     {
18         $this->stmt = $this->dbh->prepare('SELECT * FROM siswa');
19         $this->stmt->execute();
20         return $this->stmt->fetchAll(PDO::FETCH_ASSOC);
21     }
22 }
```

Kita akan coba bedah satu persatu kodenya. Pertama, kita akan buat ketika model ini diinisialisasi maka otomatis kita akan menyambungkannya dengan *database* kita dengan secara otomatis memanggil fungsi **__construct()**. Untuk hostnya kita bisa pakai 127.0.0.1 atau localhost, jika localhost tidak bisa, maka dapat lewat alamat ip localhost yaitu 127.0.0.1, misalkan localhost kalian ada port khusus seperti 8080 maka tuliskan juga portnya. Saat kita menginisialisasi PDO() maka parameternya adalah pertama DSN (*Data Source Name*), lalu *username*, dan *password*. Untuk phpMyAdmin yang menggunakan **XAMPP** sebagai web server maka *username defaultnya* adalah **root** dan *passwordnya* kosongan saja.

Lalu, untuk fungsi **getAllBlog()** disini kita mulai melakukan proses pengambilan data. Penjelasan sebagai berikut

Query Data

```
$this->stmt = $this->dbh->prepare('SELECT * FROM blog');
```

Eksekusi Query

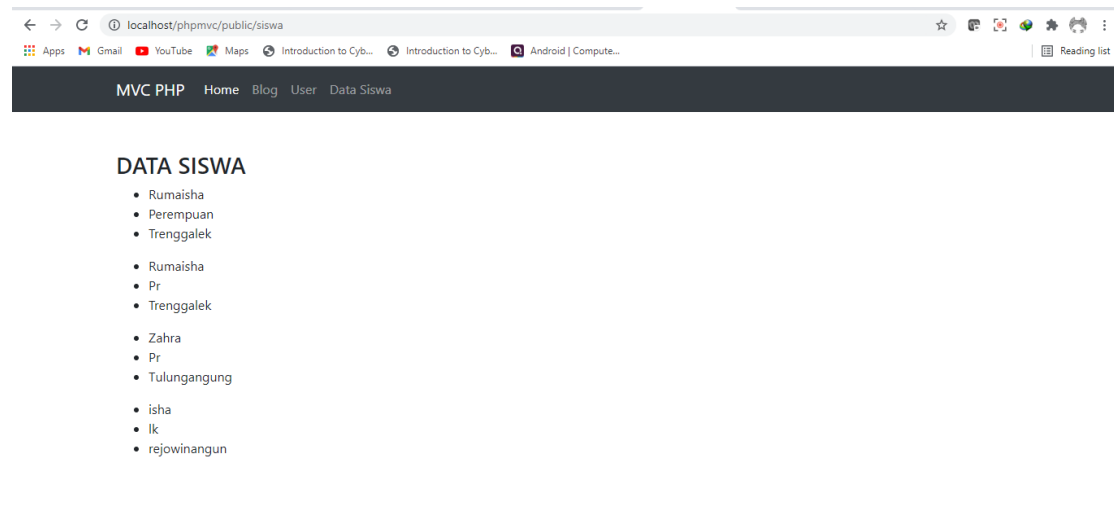
```
$this->stmt->execute();
```

Mengembalikan respon hasil query berupa array associative

```
return $this->stmt->fetchAll(PDO::FETCH_ASSOC);
```

Jika merasa proses penggunaan PDO ini lebih ribet dan sedikit agak kompleks dibanding mysql itu memang benar, dikarenakan banyaknya step itu menandakan driver PDO ini sangat berhati-hati dalam melakukan koneksi ke *database* agar datanya tetap aman dan oke.

Hasil dari Model yang dipanggil dari *database* Adalah



b. Database Wrapper

Selanjutnya kita akan coba menfokuskannya untuk pengelolaan data di *database* dengan **Database Wrapper**, yaitu berupa *class* yang menangani proses koneksi dan *query* ke *database* yang kita bisa terapkan *class* ini pada berbagai **Model**. Untuk materi yang sebelumnya, kita sebenarnya sudah melakukan pemrosesan data di *database*, walaupun hanya sebatas membaca datanya saja. Akan tetapi bentuk koneksi data ke *database* yang menjadi satu dengan model ini dirasa kurang baik, selain karena kurang efektif (kita harus menulis ulang kode koneksi dan *query*) juga kode yang telah kita tulis sebelumnya masih bisa dikatakan rawan terkena serangan jahat salah satunya *SQLInjection*.

Buat konfigurasi *database* terpisah dari model

Hal yang akan kita lakukan hampir sama ketika kita membuat *BASE_URL* sebagai url dasar dan bisa kita panggil dimana saja. Nah, kali ini kita akan menggabungkan semua variabel konstan ke dalam satu file **config**, agar kita mengurangi potensi user mengutak-atik bagian dalam dari web kita tak terkecuali, konfigurasi *database*.

1) Copy define dari *BASE_URL* dan hapus file **Constants.php**

2) Buat file config pada **app/config/config.php**

Edit config.php

```
app > config > config.php > ...
1  <?php
2  define('BASE_URL', "http://localhost/phpmvc/public");
3  //DB
4  define('DB_HOST', 'localhost');
5  define('DB_USER', 'root');
6  define('DB_PASS', '');
7  define('DB_NAME', 'phpmvc');
```

Kita definisikan setiap *variable constant* kita sesuai konfigurasi *databasenya*

```
DB_HOST : hosting dari web kita

DB_USER : username

DB_PASS : password

DB_NAME : nama database
```

Lalu mengedit file **init.php** karena constants.php diganti dengan **config.php**

Edit init.php

```
app > init.php
1  <?php
2  require_once 'core/App.php';
3  require_once 'core/Controller.php';
4  require_once 'config/config.php';
```

3) Buat Database.php sebagai Database Wrapper

Pertama kita buat dulu file **Database.php** di folder **core**, kemudian tambahkan kode seperti di bawah ini

```
<?php
class Database{
    private $host = DB_HOST;
    private $user = DB_USER;
    private $pass = DB_PASS;
    private $db_name = DB_NAME;
    private $dbh;
    private $stmt;
    public function __construct()
    {
```



```

//data source name
$dsn = "mysql:host=" . $this->host . ";dbname=" . $this->db_name;
$option = [
    PDO::ATTR_PERSISTENT => true,
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
];
try {
    $this->dbh = new PDO($dsn, $this->user, $this->pass, $option);
} catch (PDOException $e) {
    die($e->getMessage());
}
}
public function query($query)
{
    $this->stmt = $this->dbh->prepare($query);
}
public function bind($param, $value, $type = null)
{
    if (is_null($type)) {
        switch (true) {
            case is_int($value):
                $type = PDO::PARAM_INT;
                break;
            case is_bool($value):
                $type = PDO::PARAM_BOOL;
                break;
            case is_null($value):
                $type = PDO::PARAM_NULL;
                break;
            default:
                $type = PDO::PARAM_STR;
                break;
        }
    }
    $this->stmt->bindValue($param, $value, $type);
}
public function execute()
{

```

```

        $this->stmt->execute();
    }
    public function resultAll()
    {
        $this->execute();
        return $this->stmt->fetchAll(PDO::FETCH_ASSOC);
    }
    public function resultSingle()
    {
        $this->execute();
        return $this->stmt->fetch(PDO::FETCH_ASSOC);
    }
}

```

```

public function __construct() {
    //data source name

    $dsn = "mysql:host=". $this->host.";dbname=" . $this->db_name;

    $option = [
        PDO :: ATTR_PERSISTENT => true,
        PDO :: ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION

    ];

    try {
        $this->dbh = new PDO($dsn, $this->user,$this->pass, $option);
    } catch(PDOException $e) {
        die($e->getMessage());
    }
}

```

Lalu, dibagian **__construct()** kita mulai jalankan koneksi dengan *databasenya*, kita pertama buat dulu **DSN** dari db-nya lalu kita buat **option** untuk mengoptimasi kerja dari PDO kita, PDO::ATTR_PERSISTENT untuk membuat *database* kita koneksinya terjaga dan PDO::ATTR_ERRMODE yang kita *set valuenya*, PDO::ERRMOD_EXCEPTION untuk menghindari jika ada *error* yang muncul tidak membuat web kita *crash*.

Setelah itu, kita *try catch* untuk menjalankan koneksinya dengan *database*. Jika dilihat, memang proses menggunakan PDO ini rumit tapi disini lain membuat web kita lebih teroptimasi dan juga lebih *secure*.

```
public function query($query) {  
  
    $this->stmt = $this->dbh->prepare($query);  
  
}
```

Selanjutnya kita buat fungsi untuk menjalankan querynya. Pada PDO ini sebelum kita mengeksekusi sebuah query kita gunakan **fungsi prepare()** terlebih dahulu. Hal ini agar memastikan query kita benar-benar aman.

Fungsi **bind()** ini untuk mengolah atau mengidentifikasi terlebih dahulu tipe data dari parameter pada query yang dikirimkan bila ada suatu parameter dalam query kita, misalnya ketika kita melakukan query WHERE id=.. SET id= .. nah, agar lagi-lagi menjaga keamanan dari web kita, dan memastikan bahwa parameter yang dimasukan benar-benar relevan dengan query yang akan dijalankan.
public function execute() {

```
$this->stmt->execute();  
  
}  
  
public function resultAll() {  
  
    $this->execute();  
  
    return $this->stmt->fetchAll(PDO::FETCH_ASSOC);  
  
}  
  
public function resultSingle() {  
  
    $this->execute();  
  
    return $this->stmt->fetch(PDO::FETCH_ASSOC);  
  
}
```

Setelah itu kita buat fungsi untuk mengeksekui dan mengembalikan datanya, **resultAll()** untuk mengambil keseluruhan data dan **resultSingle()** untuk mengambil satu data saja.

4) Tambahkan Database.php pada init.php

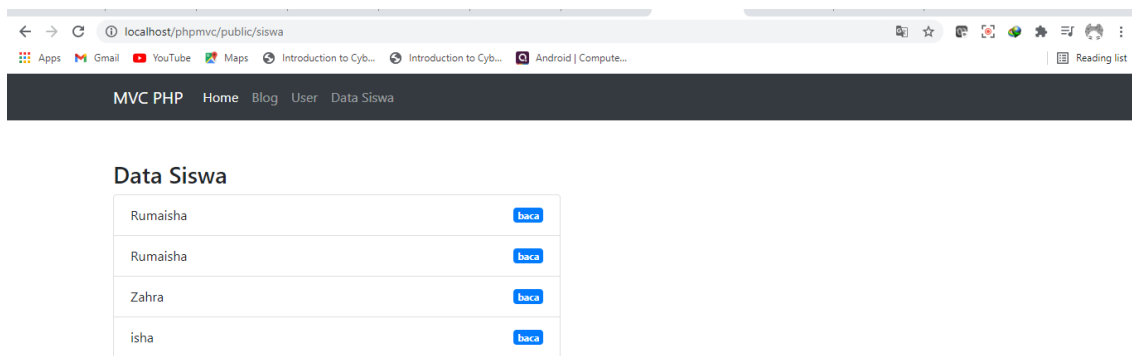
```
app > init.php
1  <?php
2  require_once 'core/App.php';
3  require_once 'core/Controller.php';
4  require_once 'config/config.php';
5  require_once 'core/Database.php';
```

5) Olah database di Model (Siswa_model.php)

```
app > models > Siswa_model.php > PHP Intelephense > Siswa_model
1  <?php
2  class Siswa_model
3  {
4      private $table = 'siswa';
5      private $db;
6
7      public function __construct()
8      {
9          $this->db = new Database;
10     }
11
12     public function getAllBlog()
13     {
14         $this->db->query("SELECT * FROM " . $this->table);
15         return $this->db->resultAll();
16     }
17     public function getBlogById($id)
18     {
19         $this->db->query("SELECT * FROM " . $this->table . 'WHERE id=:id');
20         $this->db->bind('id', $id);
21         return $this->db->resultSingle();
22     }
23 }
```

Kita rombak ulang Model kita, pertama kita buat dulu *variable* properti untuk *table* dan *db* kita. Lalu kita buat setiap Model ini dipanggil, secara otomatis **class Database** akan diinstansiasi sehingga kita dapat menggunakan *method-method* yang ada di dalamnya, setelah itu kita buat method **getAllBlog()** untuk mengambil semua data siswa, dan **getBlogById()** untuk mengambil data detail dari tiap tulisan pada siswa.

Memperbaiki tampilan Data siswa



6) Edit view pada siswa/index.php

```
<div class="container mt-5">
  <div class="row">
    <div class="col-6">
      <h3>Data Siswa</h3>
      <ul class="list-group">
        <?php foreach ($data['siswa'] as $siswa) : ?>
          <li class="list-group-item list-group-item d-flex
            justify-content-between align-items-center">
            <?= $siswa['nama']; ?>
            <a href="<?= BASE_URL; ?>/siswa/detail/<?= $siswa['id']; ?>"
              class="badge badge-primary">baca</a>
          </li>
        <?php endforeach; ?>
      </ul>
    </div>
  </div>
</div>
```

Tambahkan method detail pada *controller Siswa.php*

```
public function detail($id)
{
    $data['judul'] = "Detail Siswa";
    $data['siswa'] = $this->model("Siswa_model")->getBlogById($id);
    $this->view('templates/header', $data);
    $this->view('siswa/detail', $data);
    $this->view('templates/footer');
```

Kita ambil data id detail siswanya melalui link yang ada, kemudian dengan bekal id itu kita ambil data dari *database* dengan memanggil method `getBlogById()`. Kemudian kita arahkan body content-nya ke view baru yaitu siswa/*detail.php*.

Buat baru di view/siswa/detail.php

```

<div class="container mt-5">
  <div class="card" style="width: 18rem;">
    <div class="card-body">
      <h5 class="card-title"><?php echo $data['siswa']['nama']; ?></h5>
      <h6 class="card-subtitle mb-2 text-muted"><?php echo $data['siswa']['jenis_kelamin']; ?></h6>
      <p class="card-text"><?php echo $data['siswa']['alamat']; ?></p>
      <a href="<?php echo BASE_URL; ?>/siswa" class="card-link">Kembali</a>
    </div>
  </div>
</div>

```

c. Insert

Pada pembahasan ini kita akan memasuki olah data secara CRUD (*Create, Read, Update, dan Delete*) dan pada tulisan ini saya akan mencoba mengambil segmen **Create/Insert Database**.

1) Buat Button Tambah pada siswa/index.php

Disini kita akan tambahkan sebuah button tambah yang ketika di-click akan mentrigger method **tambah()** yang nanti akan kita buat di controller siswa ini

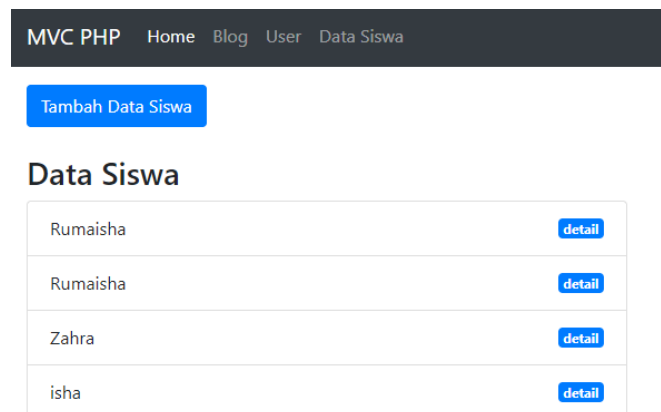
```

<div class="container mt-3">
  <div class="row">
    <div class="col-6">
      <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#formModal">
        Tambah Data Siswa
      </button>

      <h3 class="mt-4">Data Siswa</h3>
      <ul class="list-group">
        <?php foreach ($data['siswa'] as $siswa) : ?>
          <li class="list-group-item list-group-item d-flex
            justify-content-between align-items-center">
            <?php echo $siswa['nama']; ?>
            <a href="<?php echo BASE_URL; ?>/siswa/detail/<?php echo $siswa['id']; ?>" class="badge
              badge-primary">detail</a>
          </li>
        <?php endforeach; ?>
      </ul>
    </div>
  </div>
</div>

```

Sehingga tampilannya seperti dibawah ini



2) Buat Modal untuk form Data siswa

Setelah itu kita ingin buat form tambah artikel-nya itu berupa popup saja, atau dalam istilah bootstrap disebut *component* **Modal**, kita akan gunakan. Pada bagian *body* modal-nya kita akan isi dengan form sederhana. Maka dari itu kita tambahkan kode untuk Modalnya seperti ini atau bisa download dokumen pada bootstrap.

```
<!-- Modal -->
<div class="modal fade" id="formModal" tabindex="-1" role="dialog" aria-
labelledby="judulModal" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="judulModal">Tambah Data Siswa</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form action="<?= BASE_URL; ?>/siswa/tambah" method="post">

          <div class="form-group">
            <label for="nama">Nama</label>
            <input type="text" class="form-control" id="nama" name="nama">
          </div>
          <div class="form-group">
            <label for="jenis_kelamin">Jenis Kelamin</label>
            <select class="form-control" id="jenis_kelamin" name="jenis_kelamin">
              <option value="laki-laki">Laki - laki</option>
              <option value="perempuan">Perempuan</option>
            </select>
          </div>
          <div class="form-group">
            <label for="alamat">Alamat</label>
            <input type="text" class="form-control" id="alamat" name="alamat">
          </div>
        </div>
        <div class="modal-footer">
```

```

<button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
<button type="submit" class="btn btn-primary">Tambah Data</button>
</div>
</div>
</div>
</div>

```

Pastikan id dan name dari setiap component yang kita buat untuk view-ini sudah sesuai, sebab itu sangat erat hubungannya ketika kita mengirmkan datanya ke method **tambah()**. Apabila sudah, maka viewnya akan tampak seperti berikut ini.

3) Tambahkan method tambah() di Controller Blog

```

public function tambah()
{
    if ($this->model('Siswa_model')->tambahData($_POST) > 0) {
        header('Location: ' . BASE_URL . '/siswa');
        exit;
    }
}

```

Di dalam model Siswa_model kita juga akan menambahkan method tambahData() yang akan menginsert kan datanya ke *database* yang kemudian akan mereturn jumlah baris yang diperbarui, bila nilainya 1, itu artinya data berhasil ditambahkan, selain itu barangkali ada kegagalan dalam menambah data. Lalu, ketika data sudah berhasil *website* kita juga diarahkan kembali ke halaman awal view /siswa.

4) Tambahkan method `buarArtikel()` di `Blog_model.php`

```
public function tambahData($data)
{
    $query = " INSERT INTO siswa VALUES
    ('', :nama, :jenis_kelamin, :alamat ) ";
    $this->db->query($query);
    $this->db->bind('nama', $data['nama']);
    $this->db->bind('jenis_kelamin', $data['jenis_kelamin']);
    $this->db->bind('alamat', $data['alamat']);
    $this->db->execute();
    return $this->db->rowCount();
}
```

Sesuai prosedur PDO di *Database Wrapper* yang kita buat, sebelum kita mengeksekusi sebuah *query* kita akan mem-*prepare* kannya dulu dengan membind-parameter-parameter dan values yang akan ditambahkan, agar tipenya dapat diketahui dengan jelas sehingga mengurangi terjadi serangan jahat. Setelah dieksekusi, kita return jumlah baris yang telah diperbarui. Jika data, berhasil ditambahkan, maka nilai yang direturn harusnya bernilai 1, selain itu ada kemungkinan bahwa data gagal ditambahkan. Nah, ada yang aneh bukan? Kita belum membuat fungsi `rowCount()` dalam *database wrapper* kita, maka dari itu kita buat dulu.

5) Tambahkan method `rowCount()` pada `Database.php`

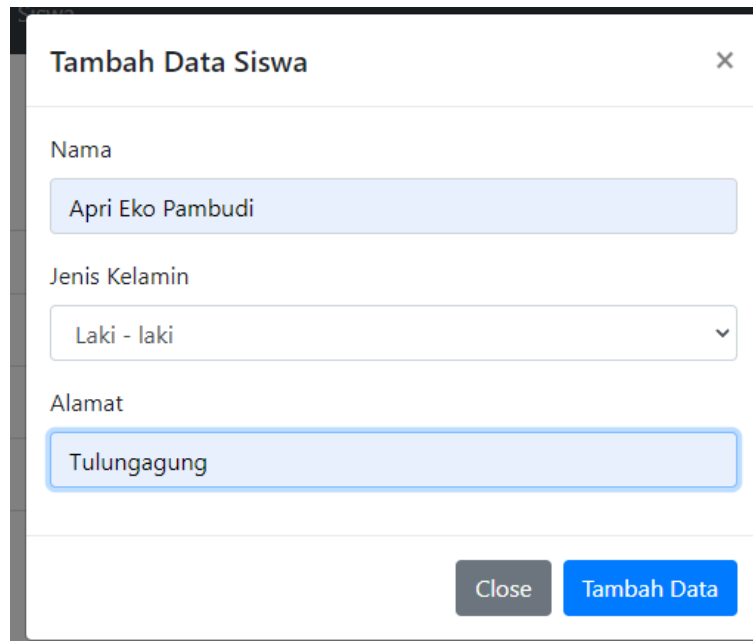
```
public function rowCount()
{
    return $this->stmt->rowCount();
}
```

Disini kita akan mengembalikan method `rowCount()` milik PDO yang menandakan ada perubahan data atau tidak. Bila ada, maka return akan bernilai 1, jika tidak bisa jadi ada kegagalan dalam menambahkan data

Data Siswa

Apri Eko Pambudi

[detail](#)



d. Flash Message

Dalam proses CRUD di sebuah web, biasanya terdapat sebuah message yang akan memberitahukan keberhasilan dari setiap perubahan akan aksi yang telah kita lakukan. Aksi itu dapat berupa create, update, maupun delete, dan untuk mengimplementasikan hal tersebut kita akan menggunakan Flash Message dengan menggunakan `$_SESSION` sebagai pelaksananya

SESSION ini adalah variabel *superglobal* yang tersimpan dalam *server* kita. Beberapa studi kasus penggunaan session misalnya, ketika kita login di sebuah *website*, dan kita jalankan SESSION untuk menyimpan data login kita, maka ketika kita membuka tab baru, dan kita buka *website* yang sama maka kita tidak perlu login ulang karena *website* sudah mendeteksi adanya session yang menyimpan status login kita. Contoh lain, misalnya dalam toko online kita menyimpan daftar belanjaan ke dalam keranjang belanja dan pada saat masuk halaman pembayaran data belanjaan tersebut masih ada. Pada flash message kita kali ini, sebenarnya kita akan menerapkannya secara lebih sederhana lagi, message hanya akan muncul sekali saja, bila web di refresh maka isi session sudah kosong.

1) Buat class baru di folder core beri nama Flasher.php

```

app > core > Flasher.php > ...
1  <?php
2
3  class Flasher {
4      public static function setFlash($pesan, $aksi, $tipe)
5      {
6          $_SESSION['flash'] = [
7              'pesan' => $pesan,
8              'aksi'  => $aksi,
9              'tipe'  => $tipe
10         ];
11     }
12
13     public static function flash()
14     {
15         if( isset($_SESSION['flash']) ) {
16             echo '<div class="alert alert-' . $_SESSION['flash']['tipe'] . ' alert-dismissible fade show" role="alert">
17                 Data Mahasiswa <strong> ' . $_SESSION['flash']['pesan'] . '</strong> ' . $_SESSION['flash']['aksi'] . '
18                 <button type="button" class="close" data-dismiss="alert" aria-label="Close">
19                 <span aria-hidden="true">&times;</span>
20                 </button>
21             </div>';
22             unset($_SESSION['flash']);
23         }
24     }
25 }

```

Pada method setFlash() kita menset data yang akan kita tampilkan di dalam flash message kita nanti, parameter pesan berisi “berhasil/gagal” nya sebuah aksi. Aksi bisa berupa menambah data, mengedit data, atau menghapus data, dan tipe nantinya kita sesuaikan dengan class bootstrap yang akan menjadi tema dari flash message kita. Kemudian pada fungsi flash() kita outputkan flash message yang telah kita buat setelah user melakukan aksi. Kenapa menggunakan static function ? Hal ini dikarenakan agar kita tidak perlu menginstansiasi class Flasher untuk dapat mengakses method-method di dalamnya.

Disini kita menggunakan class Alert pada Bootstrap dan kita memodifikasinya bila tipe: success, maka alert berwarna hijau, jika tipe: danger, maka alert berwarna merah. Kemudian pesan akan berisi “berhasil/gagal” suatu aksi dilakukan, dan pada aksi berisi suatu aksi yang dilakukan user bisa berupa penambahan data, pengeditan data, atau penghapusan data. Setelah message ditampilkan, maka semua data dari SESSION akan dihapus dengan cara unset().

2) Daftarkan Flasher.php pada init.php

```

app > init.php
1  <?php
2
3  require_once 'core/App.php';
4  require_once 'core/Controller.php';
5  require_once 'core/Database.php';
6  require_once 'core/Flasher.php';
7  require_once 'config/config.php';

```

3) Panggil Flasher dalam index.php nya siswa.

```
<div class="row">
  <div class="col-lg-6">
    <?php Flasher::flash(); ?>
  </div>
</div>
```

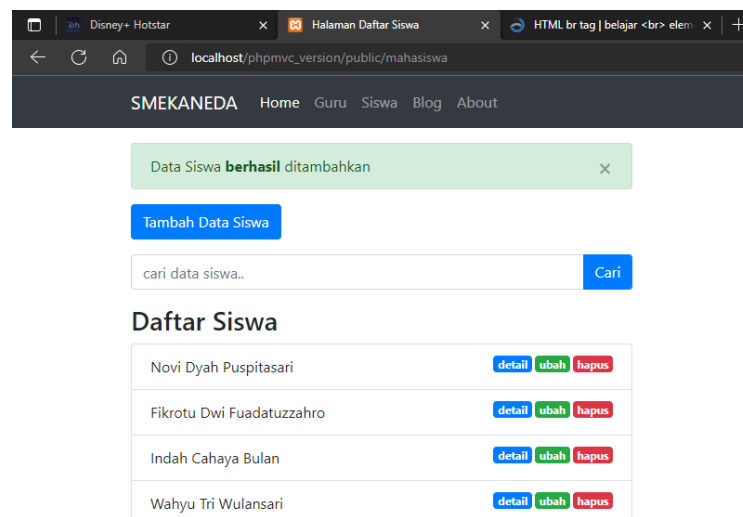
4) Edit fungsi tambah() pada Controller Siswa

```
public function tambah()
{
    if( $this->model('Siswa_model')->tambahDataSiswa($_POST) > 0 ) {
        Flasher::setFlash('berhasil', 'ditambahkan', 'success');
        header('Location: ' . BASEURL . '/siswa');
        exit;
    } else {
        Flasher::setFlash('gagal', 'ditambahkan', 'danger');
        header('Location: ' . BASEURL . '/siswa');
        exit;
    }
}
```

5) Jika ada data pada SESSION, Aktifkan Session saat public/index.php dijalankan.

```
public > index.php
1  <?php
2  if( !session_id() ) session_start();
3
4  require_once '../app/init.php';
5
6  $app = new App;
```

6) Hasil



e. Delete Data

1) Tambahkan tombol delete pada index.php nya siswa

```
<div class="row">
  <div class="col-lg-6">
    <h3>Daftar Siswa</h3>
    <ul class="list-group">
      <?php foreach( $data['siswa'] as $siswa ) : ?>
        <li class="list-group-item">
          <?= $siswa['nama']; ?>
          <a href="<?= BASEURL; ?>/siswa/hapus/<?= $siswa['id']; ?>" class="badge badge-
danger float-right" onclick="return confirm('yakin?');">hapus</a>
          <a href="<?= BASEURL; ?>/siswa/ubah/<?= $siswa['id']; ?>" class="badge badge-
success float-right tampilModalUbah" data-toggle="modal" data-target="#formModal" data-
id="<?= $siswa['id']; ?>">ubah</a>
          <a href="<?= BASEURL; ?>/siswa/detail/<?= $siswa['id']; ?>" class="badge badge-
primary float-right">detail</a>
        </li>
      <?php endforeach; ?>
    </ul>
  </div>
</div>
```

onclick="return... digunakan untuk memberikan notifikasi atau alert penghapusan data, fungsinya jika tidak sengaja tombol hapus ter klik, maka data tidak akan langsung hilang.

2) Buka controller/Siswa.php untuk menambahkan method hapus (bisa mengcopy dari method tambah).

```
public function hapus($id)
{
    if( $this->model('Siswa_model')->hapusDataSiswa($id) > 0 ) {
        Flasher::setFlash('berhasil', 'dihapus', 'success');
        header('Location: ' . BASEURL . '/siswa');
        exit;
    } else {
        Flasher::setFlash('gagal', 'dihapus', 'danger');
        header('Location: ' . BASEURL . '/siswa');
        exit;
    }
}
```

3) Kemudian buka folder model/Siswa_model.php

```
public function hapusDataSiswa($id)
{
    $query = "DELETE FROM data_siswa WHERE id = :id";

    $this->db->query($query);
    $this->db->bind('id', $id);

    $this->db->execute();

    return $this->db->rowCount();
}
```

4) Hasil

SMEKANEDA Home Guru Siswa Blog About

Data Siswa **berhasil** dihapus

Tambah Data Siswa

cari data siswa.. Cari

Daftar Siswa

| | |
|---------------------------|---|
| Novi Dyah Puspitasari | detail ubah hapus |
| Fikrotu Dwi Fuadatu Zahro | detail ubah hapus |
| Indah Cahaya Bulan | detail ubah hapus |

f. Update Data

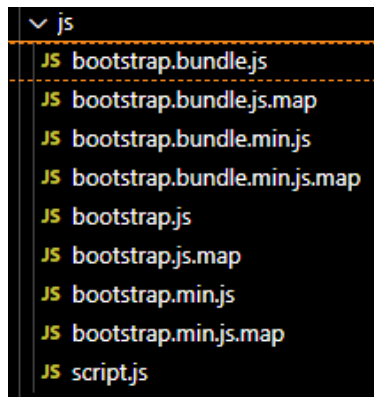
1) Tambahkan tombol update pada index.php nya siswa.

Pada bagian ini tidak perlu mengetikkan kode program untuk update data karena sudah kita buat ketika proses pembuatan tombol delete.

```
<a href="<?= BASEURL; ?>/siswa/ubah/<?= $siswa['id']; ?>" class="badge badge-success float-right tampilModalUbah" data-toggle="modal" data-target="#formModal" data-id="<?= $siswa['id']; ?>">ubah</a>
```

Data-toggle="modal" data-target="#formModal" berfungsi untuk menampilkan modal yang sama dengan tombol tambah.

2) Buat file javascript (jQuery) pada public/js/script.js



Sebelum kita isi file script.js, kita tautkan dulu file script.js ini akan dipanggil lewat view/template/footer.php. Tambah file berikut ini di bawah bootstrap.

```
<script src="<?= BASEURL; ?>/js/script.js"></script>
```

Kemudian isi script berikut ini:

```
$(function() {

    $('#tombolTambahData').on('click', function() {
        $('#formModalLabel').html('Tambah Data Siswa');
        $('.modal-footer button[type=submit]').html('Tambah Data');
        $('#nama').val("");
        $('#jenis_kelamin').val("");
        $('#alamat').val("");
        $('#id').val("");
    });

    $('#tampilModalUbah').on('click', function() {

        $('#formModalLabel').html('Ubah Data Siswa');
        $('.modal-footer button[type=submit]').html('Ubah Data');
        $('.modal-body form').attr('action', 'http://localhost/phpmvc/public/siswa/ubah');

        const id = $(this).data('id');

        $.ajax({
            url: 'http://localhost/phpmvc/public/siswa/getubah',
            data: {id : id},
            method: 'post',
            dataType: 'json',
            success: function(data) {
```

```

        $('#nama').val(data.nama);
        $('#jenis_kelamin').val(data.jenis_kelamin);
        $('#alamat').val(data.alamat);
        $('#id').val(data.id);
    }
});

});

```

3) Buat method getubah pada controller/Siswa.php

```

public function getubah()
{
    echo json_encode($this->model('Siswa_model')->getSiswaById($_POST['id']));
}

public function ubah()
{
    if( $this->model('Siswa_model')->ubahDataSiswa($_POST) > 0 ) {
        Flasher::setFlash('berhasil', 'diubah', 'success');
        header('Location: ' . BASEURL . '/siswa');
        exit;
    } else {
        Flasher::setFlash('gagal', 'diubah', 'danger');
        header('Location: ' . BASEURL . '/siswa');
        exit;
    }
}
}

```

4) Selanjutnya buka view/siswa/index.php

Pada index.php akan timbuhkan kode program ini pada bagian modal, tepatnya di bawah form action /siswa/ubah.

```
<input type="hidden" name="id" id="id">
```

5) Kemudian buka model/Siswa_model.php

Kita tambahkan kode function ubahDataSiswa seperti di bawah ini:

```

public function ubahDataSiswa($data)
{
    $query = "UPDATE siswa SET

```

```

        nama = :nama,
        jenis_kelamin = :jenis_kelamin,
        alamat = :alamat
    WHERE id = :id";

    $this->db->query($query);
    $this->db->bind('nama', $data['nama']);
    $this->db->bind('jenis_kelamin', $data['jenis_kelamin']);
    $this->db->bind('alamat', $data['alamat']);
    $this->db->bind('id', $data['id']);

    $this->db->execute();

    return $this->db->rowCount();
}

```

E. TUGAS

1. Bacalah dengan seksama materi ini. Jika ada yang kurang paham, silakan bertanya kepada guru.
2. Cobalah langkah-langkah membuat web bergaya MVC yang terdapat pada modul ini.
3. Untuk project akhir MVC-PHP, pada data Guru dan siswa isikan sesuai dengan db_smkn2 dari materi basis data.
4. Hasil akhir project web bergaya MVC sebagai berikut ini:

Project akhir = **web profile sekolah**, dimana susunannya sebagai berikut ini:

| Home | Kompetensi Keahlian | Data Guru | Data Siswa | About |
|--|---|---|--|---|
| <ul style="list-style-type: none"> • Isi dengan sambutan dan data lainnya tentang SMKN 2 Trenggalek | <ul style="list-style-type: none"> • Teknik Konstruksi dan Perumahan (TKP) • Desain Pemodelan dan Informasi Bangunan (DPIB) • Teknik Pemanasan, Tata Udara dan Pendinginan (TPTUP) • Teknik Pengelasan (TP) • Rekayasa Perangkat Lunak (RPL) • Kuliner • Akuntansi | <ul style="list-style-type: none"> • Terdapat tabel guru dan tombol untuk tambah, edit, detail dan hapus | <ul style="list-style-type: none"> • Terdapat tabel siswa dan tombol untuk tambah, edit, detail dan hapus | <ul style="list-style-type: none"> • Berisi data pembuat Website |

😊😊😊😊😊 SELAMAT BELAJAR 😊😊😊😊😊