**PORTOFOLIO**

1. **Pendahuluan**

**Portofolio** adalah sekumpulan informasi yang menunjukkan hasil karya, skill, pendidikan, pengalaman berorganisasi, pelatihan yang pernah ditempuh, dan pengalaman kerja Anda.

**Fungsi portofolio** adalah menunjukkan pengalaman kerja Anda menggunakan contoh hasil karya terbaik yang pernah dihasilkan.

Apapun jenis portofolio yang digunakan, Anda bisa lebih meyakinkan perusahaan atau calon klien Anda lewat karya nyata.

**Portofolio biasa dan Website Portofolio** Lalu apa yang membedakannya? perbedaanya adalah kemudahan mengaksesnya. Dengan website porfolio, Anda tidak perlu membawa atau menyalin portofolio fisik. Yang Anda butuhkan hanyak url website porfolio yang sudah dibuat sebelumnya untuk menunjukkan siapa Anda.

**Di era digital** seperti saat ini, memiliki website portofolio adalah hal penting karena akan memberikan kesan profesional bagi siapapun yang mengunjunginya. Selain agar terlihat profesional, membuat website portofolio juga dapat menjangkau audiens yang lebih luas.

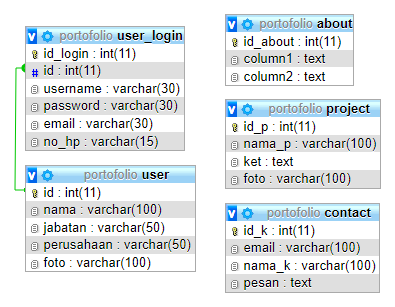
**Wesbite portofolio** adalah portofolio digital yang dapat diakses melalui internet dengan tujuan untuk menunjukan karya dan bakat Anda. Pada website yang dibuat, Anda bisa menunjukkan bakat, keahlian dan pengalaman yang telah Anda miliki.

Sangat praktis bukan? dengan hanya 1 URL Anda bisa menunjukkan kepada semua orang tentang karya dan pengalaman Anda.

1. **Pembuatan Wesbite portofolio**
2. **Kebutuhan data**

Dari hasil analisis minimal data yang dibutuhkan adalah, data diri, data project, dan data komunikasi.

Buatlah database **portofolio** dengan ketetapan sebagai berikut :



1. **Alur Program**

Aplikasi yang dibangun diperuntukan untuk umum (halaman user). Maka buatlah tampilan sebagus dan semenarik mungkin (**user interface**).

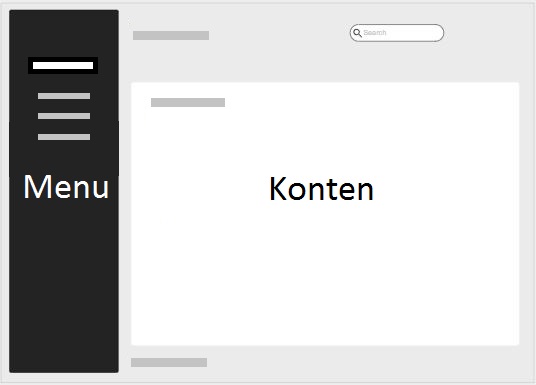
Halaman user tersebut **data**nya dikelola dari halaman pemilik (halaman admin), maka untuk halaman admin buatlah keamanan berupa **login**, dan tamilan halaman yang memfokuskan pada manajemen dan fungsi (**user experience**).



1. **Desain Aplikasi / Mockup**   
   Perancangan desain berikut dapat dikembangkan dan disempurnakan.
   * 1. **Halaman User**



* + 1. **Halaman Admin**



1. **Implementasi**
2. **Halaman User**
3. **Kebutuhan**

Sebelum melanjutkan ketahap implementasi, diharapkan sudah mengetahui dan memahami pengetahuan sebagai berikut:

* + - * + Manajemen file

Memahami pengetahuan dasar apa itu file, apa itu folder, copy, paste, cut, dan cara merapihkan tempat penyimpanan.

* + - * + Basis Data (mysql)

Mengetahui dan memahami perintah-perintah dalam DDL seperti membuat, merubah dan menghapus database dan table. Mengetahui perintah-perintah dalam DML seperti membuat, merubah, dan menghapus data.

* + - * + Html, Css, dan Javascript

Mengetahui dan Memahami tag-tag html, property-property css, dan perintah-perintah pada javascript.

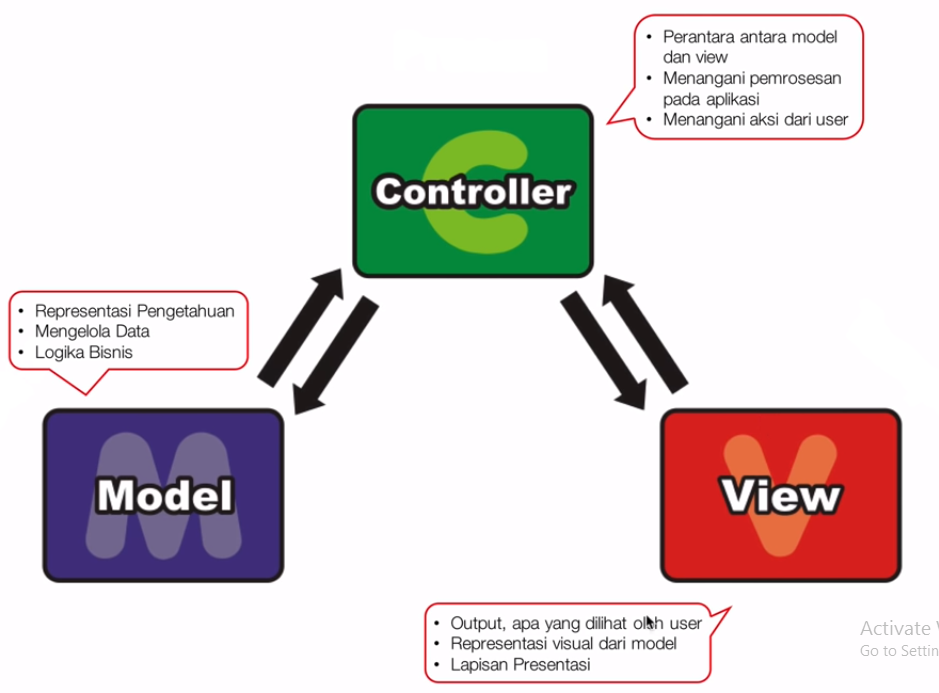
* + - * + PHP Dasar

Memahami dasar-dasar pemrograan dengan php diantaranya : variabel, tipe data, operator, perulangan, percabangan, array, fungi, dan lain sebagainya.

* + - * + Konsep OOP

Memahami konsep dasar dalam pemrograman berorientasi Objek, seperti apa itu class, property, method, constructor, inheritance, overriding dan lain sebagainya.

* + - * + Konsep MVC



Definisinya adalah pola arsitektur perancangan perangkat lunak berorientasi objek, dan salahsatu tujuannya untuk memisahkan antara **tampilan**, **data**, dan **proses.**

* + - * + Freamwork bootstrap

Framework adalah sebuah kerangka kerja yang digunakan untuk mengembangkan rekayasa perangkat lunak. Dibidang web salah satunya adalah freamwork bootstrap. Bootstrap adalah framework HTML, CSS, dan JavaScript yang berfungsi untuk mendesain website responsive dengan cepat dan mudah.

1. **Persiapan**

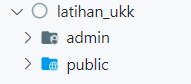
Alat/ Tools

* + - * + XAMPP (MySql dan PHP)
        + Code editor (Visual Studio Code), kemudian Install extensions PHP Intelephense (untuk membantu penggunaan syntax php).
        + Browser (Google Chrome)

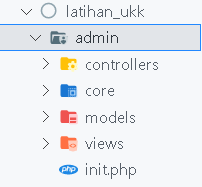
1. **Menerapkan arsitektur MVC**
2. **Persiapan Folder MVC**

Buka aplikasi Visual Studio Code, kemudian koneksikan ke folder project:

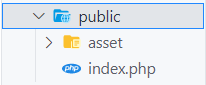
* Buatlah dua folder, yaitu **admin** dan **public** pada project portofolio teman-teman. Perhatikan gambar berikut!



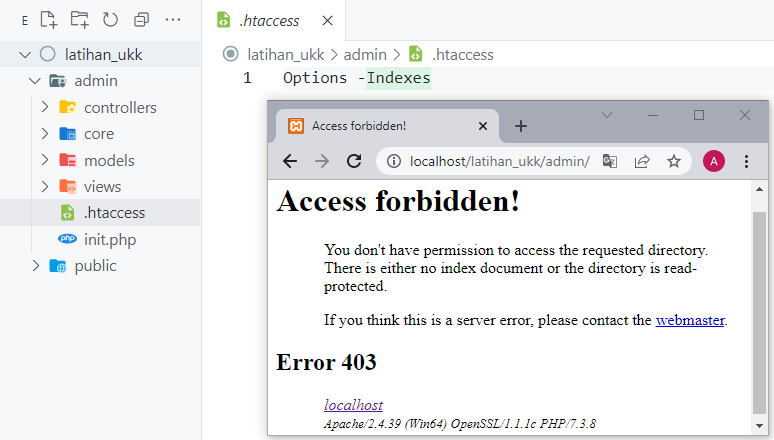
* Didalam folder **admin** buatlah 4 folder: **models, views, controllers,** dan **core** serta 1 file dengan nama **init.php**



* Di dalam folder **public**, buatlah 1 folder **asset** dan 1 file dengan nama **index.php** perhatikan gambar!



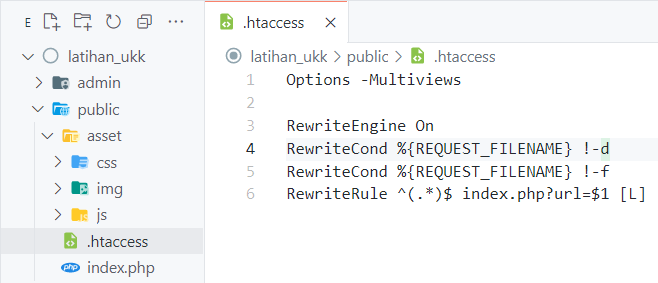
1. **Membuat Router untuk keamanan akses folder/ file**.
2. Buatlah file **.htaccess** di dalam folder admin, kemudian isikan perintah sebagai berikut :



Fungsi dari printah **Options –Indexes** adalah, ketika tidak ada file dengan nama index.php maka jangan berikan akses apapun ke file dan folder yang ada didalam folder admin. Kenapa folder admin? Karena file **.htaccess** nya ada di dalam folder admin.

1. Buat file **.htaccess** di dalam folder **public**, dengan tujuan agar url memanggil **controller** dan **method** bukan file ataupun folder.!

Jadi dari **index.php** kita akan mengakses Controller dan Method melalui file **init.php** bukan lagi folder atau file. Maka lakukan hal berikut:



|  |
| --- |
| Penjelasan :  Options –Multiviews //*untuk menghindari kesalahan dalam pemanggilan folder atau file di dalam folder public.*  RewriteEngine On // menjalankan penulis ulang url.  RewriteCond %{REQUEST\_FILENAME} !-d //jika url yang dipanggil berupa folder maka abaikan proses selanjutnya.  RewriteCond %{REQUEST\_FILENAME} !-f //jika url yang dipanggil berupa file maka abaikan proses selanjutnya.  RewriteRule ^(.\*)$ index.php?url=$1 [L] //apapun yang kita ketikan di-url akan di simpan setelah index.php?url=  [L] //jika *rule terpenuhi maka jangan jalankan rule yang lain*  Jadi sekarang setelah url <http://localhost/latihan_ukk/public/index.php> akan mengakses Controller**/**method. |

|  |
| --- |
| Kesimpulannya:  alur yang akan kita gunakan adalah ***index.php => init.php => App.php*** |

Maka isi syntax nya sebagai berikut:

**index.php**

|  |
| --- |
| <?php  //menghubungkan ke file init.php yang ada di folder admin  require\_once '../admin/init.php';  $app = new App; |

Menghubungkan ke ***init.php*** yang ada di dalam folder admin. Sebenarnya bisa menggunkana perintah *include* ataupun *require* untuk mengambil *class* *App* di file *App.php* (karena di file *init.php* menghubungkan ke file *App.php*)

**init.php**

|  |
| --- |
| <?php  //menghubungkan ke file App.php  require\_once 'core/App.php'; |

File *init.php* ini akan menjadi jembatan dari file *index.php*, dan mengatur alur dari program seperti menghuungkan ke controller/method (controller ini yang mengatur ke **view** dan **model**).

1. Buat file **App.php** yang berada di dalam folder **core**:

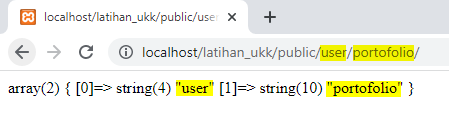
Didalam file App.php kita akan buat *parshing url* atau method yang merapihkan url. Ketikan syntaxnya sebagai berikut!

**App.php**

|  |
| --- |
| <?php  class App{      public function \_\_construct()      {          $url = $this->parseURL();          var\_dump($url);      }      public function parseURL()      {          if (isset($\_GET['url'])) {              //menghilangkan garis miring(/) di akhir url              $url = rtrim($\_GET['url'],'/');  //menghilangkar karakter aneh atau karakter yang memungkinkan kita di hack              $url = filter\_var($url, FILTER\_SANITIZE\_URL);  //menghilangkan tanda garis miring (/) dan mengambil string-nya.              $url = explode('/', $url);              return $url;          }      }  } |

Untuk menjalankan ***App.php*** alurnya adalah: *index.php* => *init.php* => *App.php*

Hasil/ Output App.php yang dijalankan melalui index.php



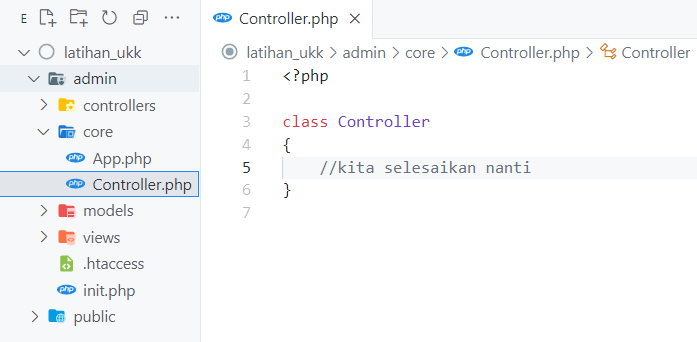
Kenapa hasilnya sebagai berikut? Karena kita menjalankan method **parseURL()** dan menampilkannya dengan **var\_dum($url)**. Ini hanya sebagai ujicoba saja bahwa konfigurasi yang kita lakukan berhasil. Nanti akan kita rubah baris tersebut!

1. **Membuat Controller dan Method Default.**

Jadi jika kita memanggil controller atau method yang tidak ada, maka program akan menjalankan controller dan method yang default. Kita akan membuat controller dan method default pada file **App.php**

Sebelum melanjutkan ke pembuatan controller dan method default, kita buat terlebih dahulu controller parent, dan controller childdengan tahapan :

* Buat file **Controller.php** di foder **core**, (controller induk / utama).



* Buat file **Portofolio.php** di foder **controller** (inilah yang akan kita jadikan *controller default*).



* Jangan lupa di file **init.php** panggil Controller yang ada di folder **core**
* **init.php**

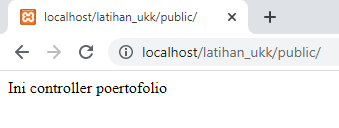
|  |
| --- |
| <?php  //menghubungkan ke file App.php  require\_once 'core/App.php';  require\_once 'core/Controller.php'; |

* Membuat controller dan method default pada file **App.php**

|  |
| --- |
| <?php  class App{      protected $controller   = 'Portofolio'; //controller default      protected $method       = 'index';      //method defaul      protected $params       = [];           //parameter jika ada      public function \_\_construct()      {          $url = $this->parseURL();          //pemanggilan controller          if (file\_exists('../admin/controllers/'.$url[0].'.php')) {              $this->controller = $url[0];              unset($url[0]);          }          require\_once '../admin/controllers/'.$this->controller.'.php';          $this->controller = new $this->controller;          //pemanggilan method          if (isset($url[1])) {              if (method\_exists($this->controller, $url[1])) {                  $this->method = $url[1];                  unset($url[1]);              }          }          //paramenters          if (!empty($url)) {              $this->params = array\_values($url);          }          //jalankan controller & method, serta kirim parameter jika ada          call\_user\_func\_array([$this->controller,$this->method],$this->params);      }      public function parseURL()      {          if (isset($\_GET['url'])) {              //menghilangkan garis miring(/) di akhir url              $url = rtrim($\_GET['url'],'/');  //menghilangkar karakter aneh atau karakter yang memungkinkan kita di hack              $url = filter\_var($url, FILTER\_SANITIZE\_URL);              //menghilangkan tanda garis miring (/) dan mengambil string-nya.              $url = explode('/', $url);              return $url;          }      }  } |

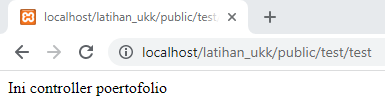
Source code yang di blok adalah tambahan source code nya

Output/ hasil ujicoba ke-1



Pada contoh diatas kita mengakses **localhost/latihan\_ukk/public/** dan hasilnya menampilkan controller Portofolio (default)

Output/ hasil ujicoba ke-2



Pada contoh kedua kita mengakses **localhost/latihan\_ukk/public/test/test** dan menampilkan hasil yang sama dengan percobaan ke 1. Karena tidak ada controller dengan nama test di dalam folder controller otomatis menjalankan controller defaultnya yaitu Portofolio.php

1. **Membuat View Portofolio**

Dari hasil praktik sebelumnya, kita sudah berhasil memanggil controller Portofolio dari file *index.php. sekarang* kita akan buat alur dari **controller Portofolio** ke folder **views**dengan tahapan sebagai berikut :

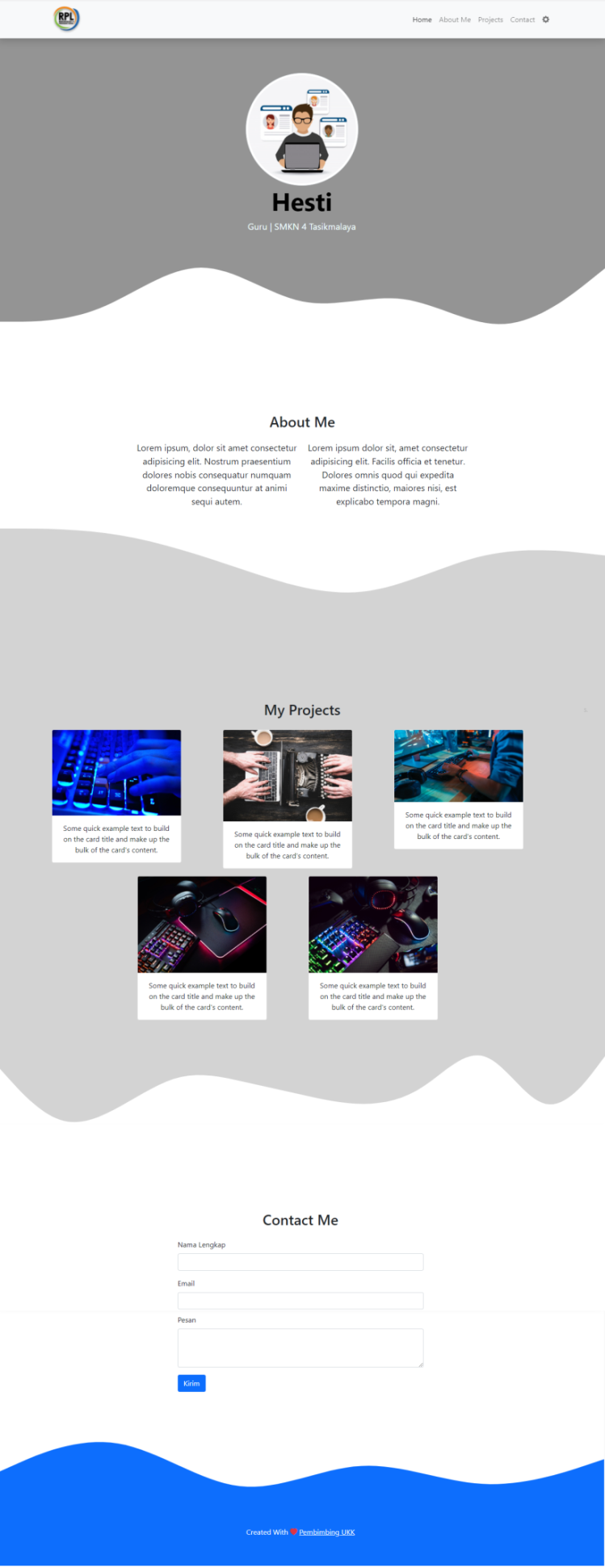
i*ndex.php=>init.php=>App.php=>Portofolio.php=>views/portofolio*

Sebelum itu siapkan dulu halaman yang akan kita jalankan nantinya.

1. **Siapkan project portofolio (halaman user)**

Pastikan **mockup** halaman user sudah di implementasikan dengan bootstrap.

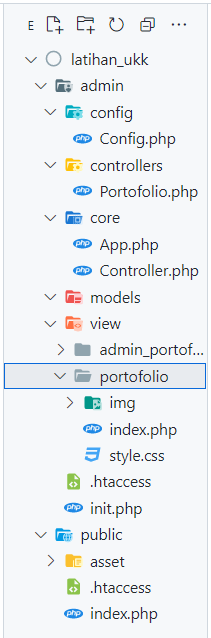
Berikut hasil implementasi yang telah dibuat:



Silahkan buat folder **portofolio** di dalam folder **views** dan **pindahkan** project portofolio yang di buat ke dalam folder **tersebut.**

Dan buat folder **admin\_portofolio** yang nantinya akan kita jadikan tampilan admin.

Perhatikan gambar berikut!

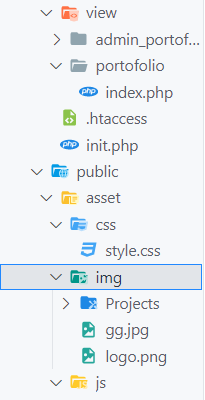


Gambar di atas adalah susunan folder **views** setelah memindahkan project portofolio. Fokus ke folder **portofolio**, di dalam folder tersebut ternyata ada folder **img** (isinya gambar yang digunakan untuk portofolio, file **index.php** (adalah tampilan utama portofolio, dan **style.php** (style yang digunakan untk tampilan portofolio terrsebut)

Lakukan hal berikut :

* pindahkan folder **img** ke folder **public/asset**
* Buat folder **css** dan **js** di folder **public/asset**
* Pindahkan **style.css** kefolder **public/asset/css**

Hasilnya sebagai berikut :



Jadi folder portofolio hanya ada **index.php**. Berhubung **img** dan **style.css** sudah di pindahkan, maka kita perlu menghubungkan ulang. Nanti saja kita lakukannya, sekarang kita akan mengkonfigurasi file controller (supaya terhubung ke ***views/portofolio/index.php***).

1. **Konfigurasi Controller.php dan Portofolio.php**

* Ketikan syntax berikut di file **Controller.php** yang ada di folder **core**.

|  |
| --- |
| <?php  class Controller  {      public function view($view, $data=[])      {          require\_once '../admin/views/'.$view.'.php';      }  } |

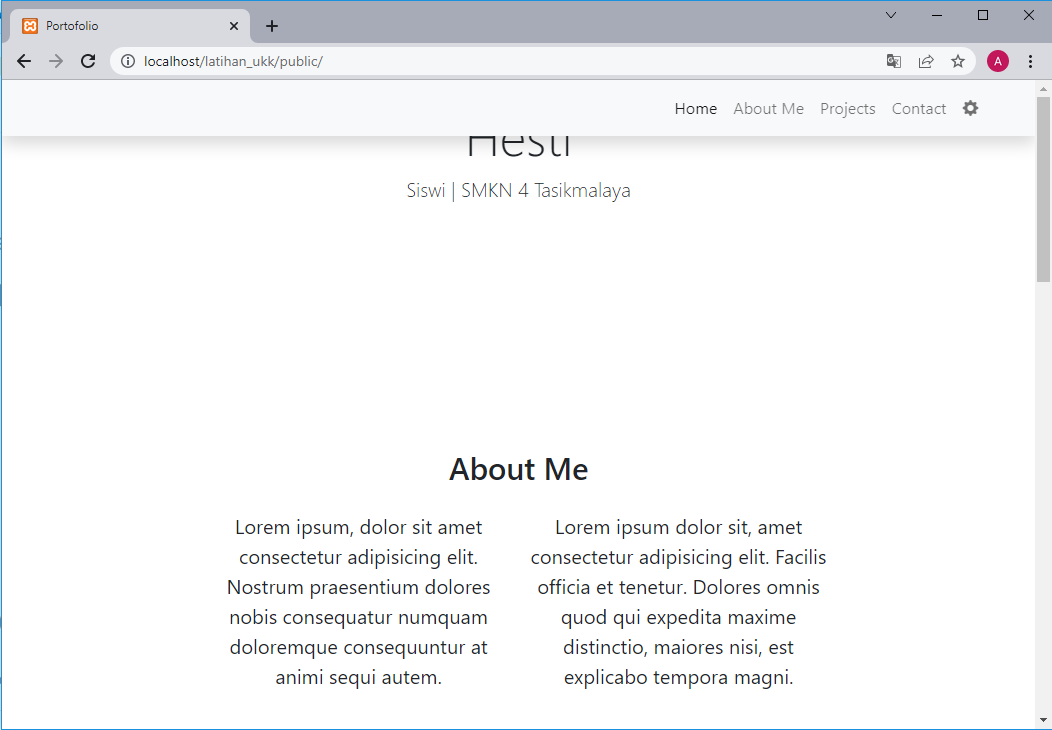
Membuat method view(), yang berfungsi mengarahkan ke folder views.

* Ketikan syntax berikut di file **Portofolio.php** yang ada di folder **controllers**

|  |
| --- |
| <?php  class Portofolio extends Controller  {      public function index()      {  $this->view('portofolio/index'); *//menargetkan ke index.php yang ada di folder views/portofolio*      }  } |

Dari controller **Portofolio.php** mengirim parameter ke method **view()** dan menjalankannya, yang bertujuan menargetkan lebih detail ke folder **views**

Output/ hasil



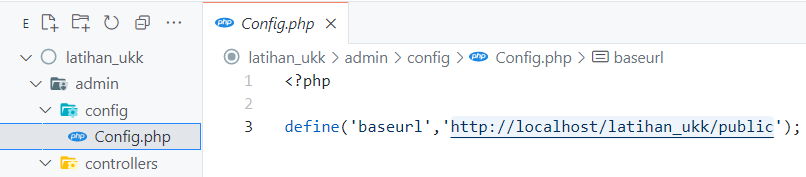
Hasil sementara tampilan nya acak-acakan, karena file style.php, dan gambar yang digunakan sudah di pindahkan ke folder asset yang ada di folder public. Oleh karena itu kita perlu menghubungkan kembali. Berikut tahapan nya!

1. **Membuat File Konfigurasi**

Kinta akan buat konstan atau *absolute url* yaitu url yang dimulai dari awal server (**http://localhost/latihan\_ukk/public**), ini akan membantu kita dalam pemanggilan **assets** seperti css dan js. File configurasi ini juga nantinya akan kita gunakan sebagai **pendefinisian *database****.* Silahkan praktikan hal-hal berikut ini:

* Buat folder **config** di dalam folder admin.
* Buat file dengan nama **Config.php** di dalam folder config

Isikan syntax sebagai berikut:



* Kemudian buka file **init.php**, dan hubungkan dengan file Config.php

init.php

|  |
| --- |
| <?php  //Menghubungkan ke App.php yang Mengatur URL  require\_once 'core/App.php';  //Menghubungkan ke Controller utama  require\_once 'core/Controller.php';  //Menghubungkan ke file konfigurasi  require\_once 'config/Config.php' |

* Buka file **index.php** yang ada di folder **views/portofolio/index.php**

Kita akan menghubungkan ulang ke **style.css** dan folder **img** dengan configurasi baseurl yang td dibuat.

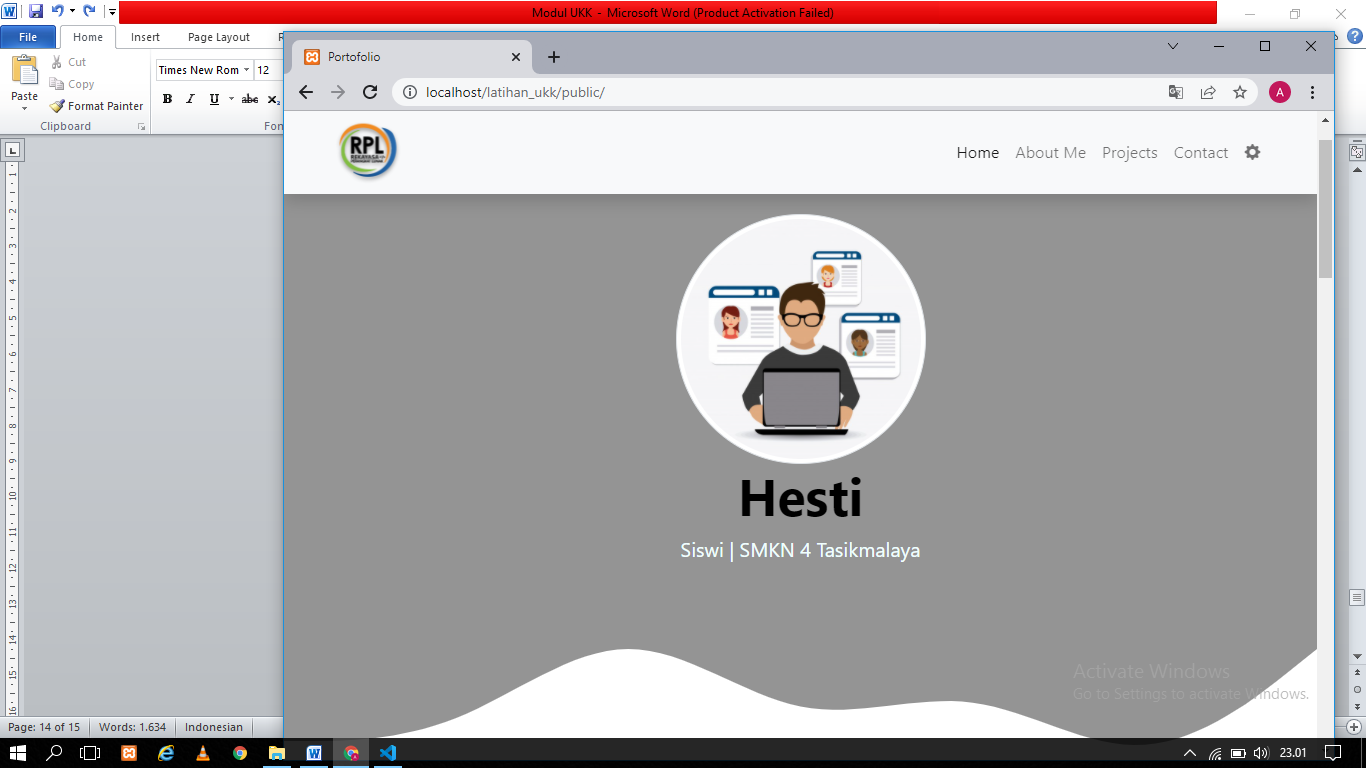
Hubungkan ulang Link ke **style.css**

|  |
| --- |
| <link rel="stylesheet" href="style.css" />  Menjadi  <link rel="stylesheet" href="<?= baseurl;?>/asset/css/style.css" /> |

Hubungkan ulang ke folder **img**

|  |
| --- |
| <img src="img/gg.jpg" alt="" class="img-thumbnail image rounded-circle" />  Menjadi  <img src="<?= baseurl;?>/asset/img/gg.jpg" alt="" class="img-thumbnail image rounded-circle" />  Lakukan hal yang sama terhadap pemanggilan gambar yang lain |

**Output/ hasil setelah dihubungkan ulang**



1. **Membuat Model**

File Model ini kita akan gunakan untuk manipulasi data dari database.

* **Require dari Controller.php ke models**

Silahkan buat **method** atau **functions** dengan nama **model()** perhatikan gambar!

    public function model($model)

    {

        require\_once '../admin/models/' . $model . '.php';

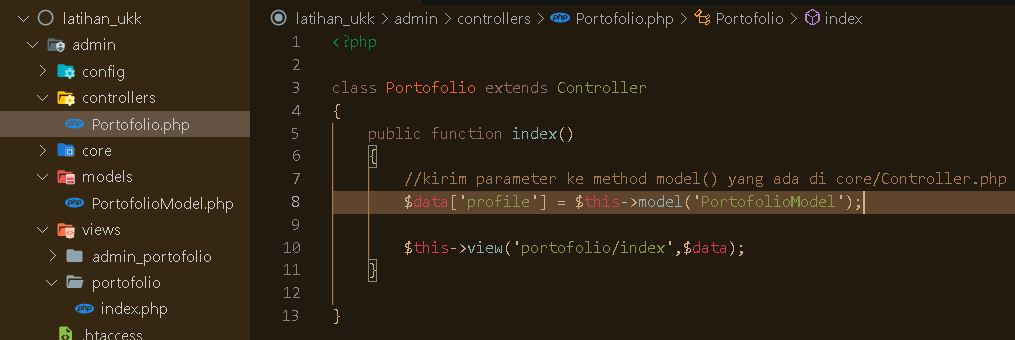
        return new $model;

    }

**$model** yang ada di method **model()** adalah sebuah parameter yang akan menampung kiriman dari file Portofolio.php

* Buka file **Portofolio.php** yang ada di dalam folder **controllers**

Di method **index()** silahkan targetkan ke **PortofolioModel.** Perhatikan gambar!



Disini kita memanggil method model() dan mengirimkan parameter **PortofolioModel** Sekarang kita nuat filenya dengan nama tersebut.

* **Membuat file model**

Silahkan buat file dengan nama **PortofolioModel.php** di dalam folder **models.** Perhatikan gambar!



Jadi kita sudah membuat file model dan sudah terhubung ke Controller.

Maka alurnya sebagai berikut :

(controllers/**Portofolio.php**) mengirim parameter ke (core**/Controller.php**) kemudian mencarinya ke=> folder models, model yang dipanggil (models**/PortofolioModel.php**) akan digunakan untuk manipulasi data dari database dengan membuat method – method kemudian nantinya method tersebut akan di panggil oleh controller dan di teruskan ke=> folder views **(**views/portofolio/**index.php)**

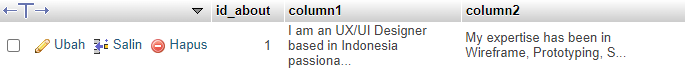
1. **Integrasi Web Portofolio dengan Database menggunakan Koneksi PDO**

Pastikan database untuk portofolio sudah dibuat. Bagi yang belum silahkan buat database **portofolio** sesuai dengan perancangan database di awal modul ini. Kemudian isi data setiap tabelnya. Perhatikan contoh :

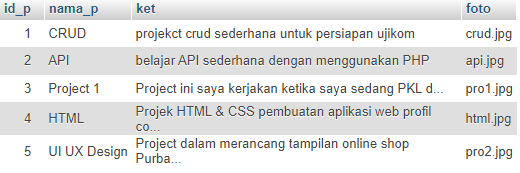
Isi tabel user



Isi tabel about



Isi tabel project



Untuk tabel contact dan user\_login biarkan kosong dulu, kita fokus di halaman user.

Silahkan praktikan tahapan nya:

1. Pendefinisan database di file **Config.php**

<?php

define('baseurl','http://localhost/latihan\_ukk/public');

//DB

define('DB\_HOST','localhost');

define('DB\_USER','root');

define('DB\_PASS','');

define('DB\_NAME','portofolio');

1. Buka file **ProtofolioModel** dan buat **koneksi** di method \_\_contstruct()

<?php

class PortofolioModel{

    private $host   = DB\_HOST;

    private $user   = DB\_USER;

    private $pass   = DB\_PASS;

    private $db\_name=DB\_NAME;

    private $dbh;

    private $stmt;

function \_\_construct()

    {

        //data source name

        $dsn = 'mysql:host='.$this->host. ';dbname='.$this->db\_name;

        $option = [

            PDO::ATTR\_PERSISTENT => true, //menjaga koneksinya terjaga

            PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION

        ];

        try {

            $this->dbh = new PDO($dsn, $this->user, $this->pass, $option);

        } catch (PDOException $e) {

            die($e->getMessage());

        }

    }

1. **Menampilkan Data User**

* Buat method di file ModelPortofolio

Untuk memampilkan data dari tabel user, kita akan buat method getProfile() dan berikut syntaknya!

public function getProfile()

    {

        $this->stmt = $this->dbh->prepare('SELECT \* FROM user');

        $this->stmt->execute();

        return $this->stmt->fetchAll(PDO::FETCH\_ASSOC);

    }

* Panggil Method di controller Portofolio.php

Setelah itu, kita panggil method getProfile() dan panggil di bagian method index() berikut syntaxnya!

public function index()

    {

       //tampung method getProfile() dari PortofolioModel.php di $data

       $data['profile'] = $this->model('PortofolioModel')->getProfile();

//kirim parameter $data ke file index.php yang ada di folder views/portofolio

       $this->view('portofolio/index',$data);

    }

* Buka dan konfigurasi views/portofolio**/index.php**

Kita panggil data yang telah di tampung pada parameter $data['profile']

<?php foreach($data['profile'] as $pro) : ?>

<!-- Jumbotron -->

 <section class="jumbotron atas text-center">

   <img src="<?= baseurl;?>/asset/img/<?= $pro['foto']; ?>" alt="" class="img-thumbnail image rounded-circle" />

   <h1 class="display-4"><?= $pro['nama']; ?></h1>

  <p class="lead"><?= $pro['jabatan']; ?> | <?= $pro['perusahaan']; ?></p>

  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1440 320">

    <path

    fill="#fff"

    fill-opacity="1"

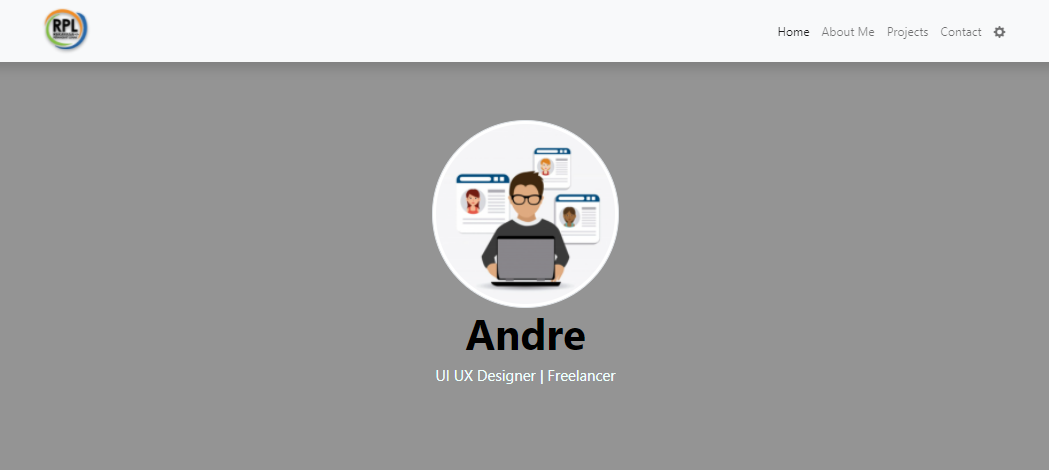
    d="M0,224L40,224C80,224,160,224,240,192C320,160,400,96,480,96C560,96,640,160,720,176C800,192,880,160,960,170.7C1040,181,1120,235,1200,229.3C1280,224,1360,160,1400,128L1440,96L1440,320L1400,320C1360,320,1280,320,1200,320C1120,320,1040,320,960,320C880,320,800,320,720,320C640,320,560,320,480,320C400,320,320,320,240,320C160,320,80,320,40,320L0,320Z"></path></svg>

</section>

<?php endforeach ?>

Pada syntax diatas yang di tandai adalah pemanggilan data field di tabel user.

Hasil/ Output



1. **Menampilkan Data About**

Perintahnya sama dengan pemanggilan data user tetapi kita sekarang mengambil data dari tabel **about**. Yang membedakan di pemanggilan method dan parameter.

* Buat method getAbout();

public function getAbout()

    {

        $this->stmt = $this->dbh->prepare('SELECT \* FROM about');

        $this->stmt->execute();

        return $this->stmt->fetchAll(PDO::FETCH\_ASSOC);

    }

* Panggil method getAbout() dan tampung pada parameter $data['about']

public function index()

    {

       //tampung method getProfile() dari PortofolioModel.php di $data

       $data['profile'] = $this->model('PortofolioModel')->getProfile();

//tampung method getAbout() kedalam $data[‘about’]

       $data['about'] = $this->model('PortofolioModel')->getAbout();

//kirim parameter $data ke file index.php yang ada di folder views/portofolio

       $this->view('portofolio/index',$data);

    }

* Panggil datanya di index.php bagian **about**.

        <div class="row justify-content-center fs-5 text-center">

          <?php foreach($data['about'] as $ab) : ?>

            <div class="col-md-4">

              <p><?= $ab['column1']; ?></p>

            </div>

            <div class="col-md-4">

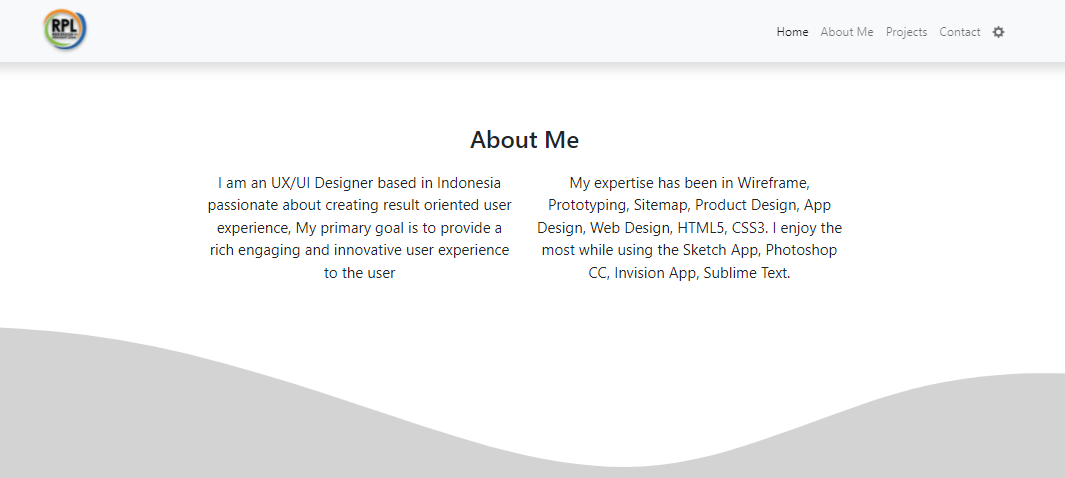
              <p><?= $ab['column2']; ?></p>

            </div>

          <?php endforeach ?>

        </div>

Output



1. **Menampilkan Data Project**

Perintahnya sama dengan pemanggilan data user tetapi kita sekarang mengambil data dari tabel **project**. Yang membedakan di pemanggilan method dan parameter.

* Buat method getProject();

public function getProject()

    {

        $this->stmt = $this->dbh->prepare('SELECT \* FROM project');

        $this->stmt->execute();

        return $this->stmt->fetchAll(PDO::FETCH\_ASSOC);

    }

* Panggil method getProject() dan tampung pada parameter $data['project']

public function index()

    {

       //tampung method getProfile() dari PortofolioModel.php di $data

       $data['profile'] = $this->model('PortofolioModel')->getProfile();

//tampung method getAbout() kedalam $data[‘about’]

       $data['about'] = $this->model('PortofolioModel')->getAbout();

//tampung method getAbout() kedalam $data[‘about’]

       $data['project'] = $this->model('PortofolioModel')->getProject();

//kirim parameter $data ke file index.php yang ada di folder views/portofolio

       $this->view('portofolio/index',$data);

    }

* Panggil datanya di index.php bagian **project**

  <div class="row justify-content-center text-center">

  <?php foreach($data['project'] as $pro) : ?>

    <div class="col-md-3 mb-3">

     <div class="card">

       <img src="<?= baseurl;?>/asset/img/projects/<?= $pro['foto']; ?>" class="card-img-top" alt="foto\_project">

       <div class="card-body">

      <h3><?= $pro['nama\_p']; ?></h3>

         <p class="card-text"><?= $pro['ket']; ?></p>

      </div>

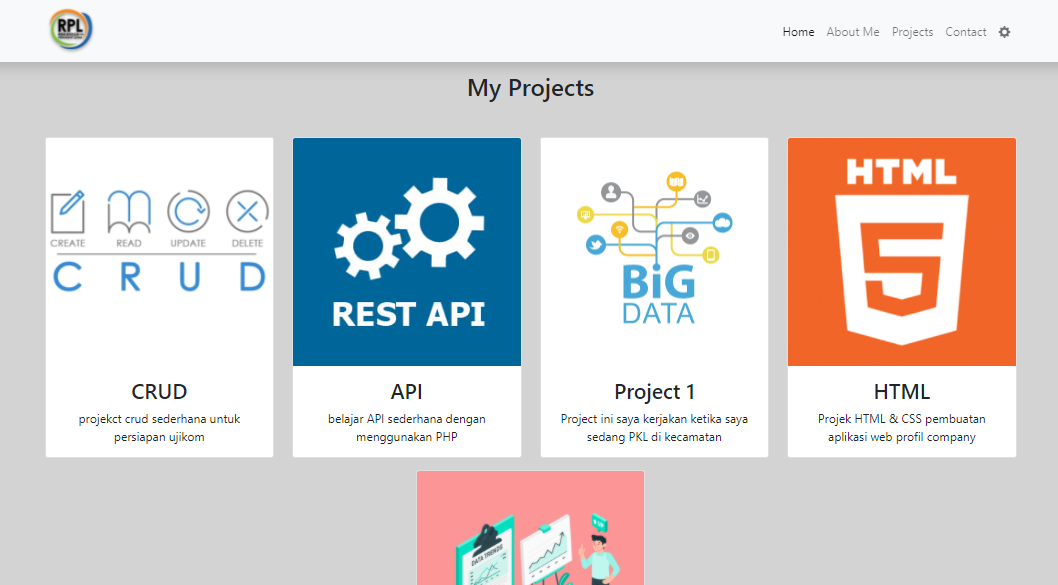
     </div>

     </div>

  <?php endforeach; ?>

  </div>

Output



1. **Mengirim Data Dari Form Contact**

* Konfigurasi form di index.php bagian **contact**

<form action="<?= baseurl;?>/Portofolio/pesan" method="POST">

     <div class="mb-3">

         <label for="name" class="form-label">Nama Lengkap</label>

         <input type="text" class="form-control" name="nama\_k" id="name" aria-describedby="name" required>

     </div>

     <div class="mb-3">

         <label for="email" class="form-label">Email</label>

         <input type="email" class="form-control" name="email" id="email" aria-describedby="email" required>

     </div>

     <div class="mb-3">

         <label for="pesan" class="form-label">Pesan</label>

         <textarea class="form-control" name="pesan" id="pesan" rows="3" required></textarea>

     </div>

     <button type="submit" class="btn btn-primary">Kirim</button>

</form>

* Buat method tampungPesan() di model PortofolioMOdel;

public function tampungPesan($data\_pesan)

    {

        $email = $data\_pesan["email"];

        $nama = $data\_pesan["nama\_k"];

        $pesan = $data\_pesan["pesan"];

        $query = "INSERT INTO contact

                VALUES

                ('','$email','$nama','$pesan')

                ";

        $this->stmt = $this->dbh->prepare($query);

        $this->stmt->execute();

        return $this->stmt->rowCount();

    }

* Buat method pesan() di Portofolio.php

public function pesan()

    {

       if ( $this->model('PortofolioModel')->tampungPesan($\_POST) > 0) {

            echo "

            <script>

            alert('Data berhasil dikirim')

            window.location.href= 'http://localhost/latihan\_ukk/public/'

            </script>

            ";

        }

        else{

            echo "gagal terkirim

            window.location.href= 'http://localhost/latihan\_ukk/public/'

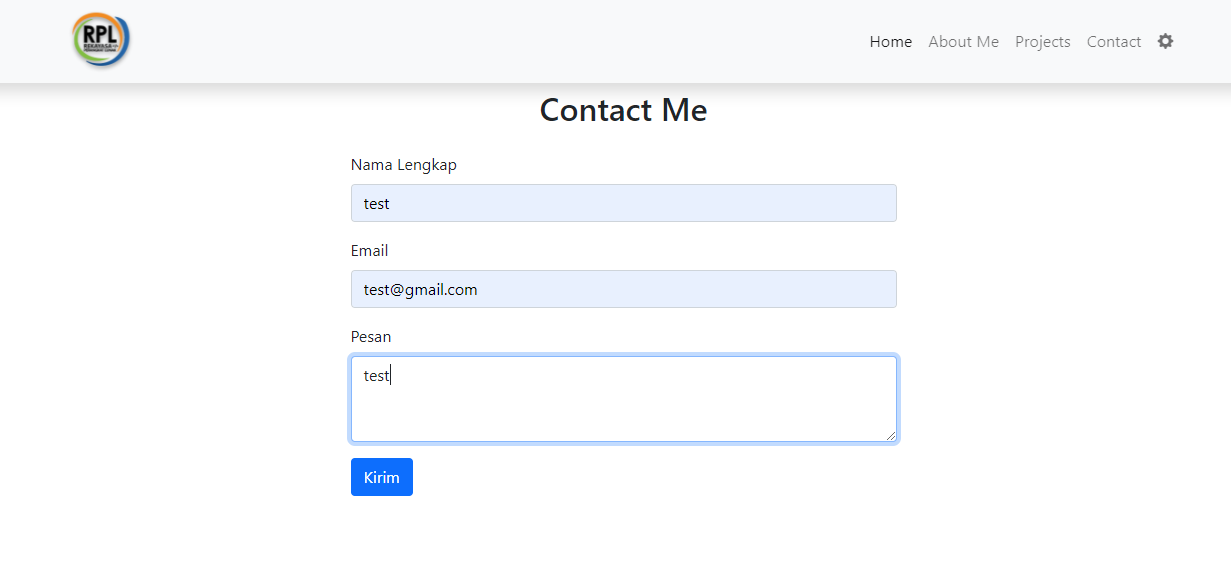
            ";

        }

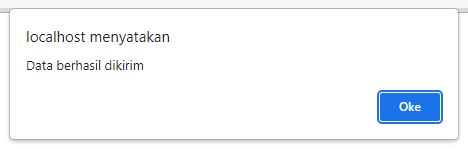
    }

Method pesan() ini di panggil melalui form yang ada di index.php

Output



Ketika di kirim, kalau berhasil ada alert yang menampilkan data berhasil dikirim, dan akan di teruskan ke halaman user (halaman utama)



1. **Halaman Admin**
2. **Koneksi dengan MySqli**
3. **Membuat halaman dashboard**
4. **Membuat Halaman Login**
5. **Membuat Update Profil**
6. **Membuat Update About/ Bio**
7. **Membuat CRUD Project**
8. **Membuat pengelolaan Contact**
9. **Hasil dan Kesimpulan**