

# Tutorial Codeigniter 4 – Part 9 – Cara Membuat CRUD dengan Codeigniter 4

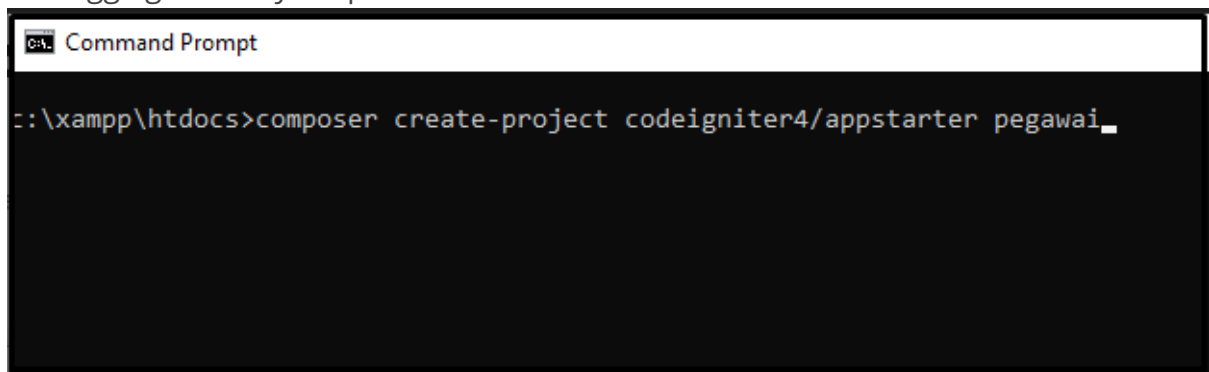
## Install Codeigniter 4

Untuk langkah awal kita perlu menginstall codeigniter 4 terlebih dahulu, dalam contoh ini kita install codeigniter 4 dalam folder dengan nama **pegawai**.

karena dalam contoh ini kita gunakan xampp, dan secara default untuk web direktori xampp berada di **C:/xampp/htdocs**, jadi silahkan masuk kedalam folder tersebut, melalui terminal, lalu berikutnya ketikkan perintah seperti gambar dibawah ini :

```
composer create-project codeigniter4/appstarter pegawai
```

Sehingga gambarnya seperti dibawah ini :



```
Command Prompt

C:\xampp\htdocs>composer create-project codeigniter4/appstarter pegawai_
```

lalu tekan enter, tunggu hingga proses instalasi selesai..

## Membuat Migration

Berikutnya kita akan membuat bagian **migration**, karena nantinya kita akan buat **tabel di database**, karena dalam contoh ini kita akan membuat crud untuk data pegawai, maka nantinya kita akan buat tabel didatabase dengan nama pegawai juga, untuk mempermudah pemahaman.

baik langkah – langkah untuk proses pembuatan migration adalah sebagai berikut :

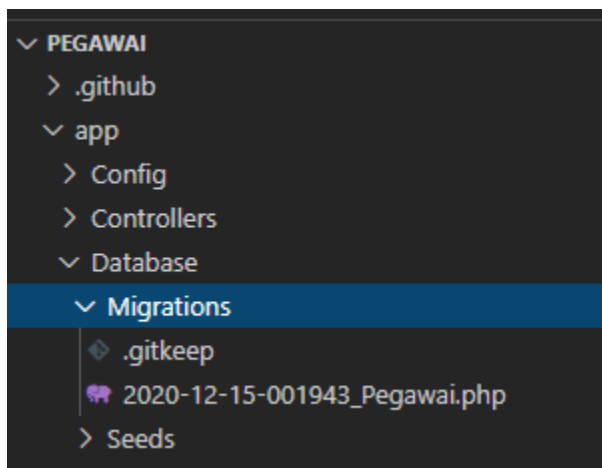
```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

C:\xampp\htdocs\pegawai>php spark migrate:create Pegawai
```

Silahkan buka terminal dan masuk kedalam direktori : **C:\Xampp\htdocs\pegawai**, lalu tuliskan perintah

```
php spark migrate:create Pegawai
```

lalu tekan enter



Lalu akan terbuat sebuah file didalam folder **app/Database/Migrations**, nama file tersebut akan diakhiri dengan nama Pegawai, buka file tersebut, dan isi dengan code seperti berikut ini :

```
<?php

namespace App\Database\Migrations;

use CodeIgniter\Database\Migration;

class Pegawai extends Migration
{
    public function up()
    {
        $this->forge->addField([
            'id_pegawai' => [
                'type'      => 'INT',
                'constraint' => 11,
                'unsigned'   => true,
```

```

        'auto_increment' => true,
    ],
    'nama' => [
        'type' => 'VARCHAR',
        'constraint' => '255',
    ],
    'jenis_kelamin' => [
        'type' => 'ENUM',
        'constraint' => "'pria','wanita'",
    ],
    'no_telp' => [
        'type' => 'VARCHAR',
        'constraint' => '100',
    ],
    'email' => [
        'type' => 'VARCHAR',
        'constraint' => '100',
    ],
    'alamat' => [
        'type' => 'VARCHAR',
        'constraint' => '255',
    ],
    'created_at' => [
        'type' => 'DATETIME',
        'null' => true,
    ],
    'updated_at' => [
        'type' => 'DATETIME',
        'null' => true,
    ]
    ];
    $this->forge->addPrimaryKey('id_pegawai');
    $this->forge->createTable('pegawai');
}

//-----

public function down()
{
    $this->forge->dropTable('pegawai');
}
}

```

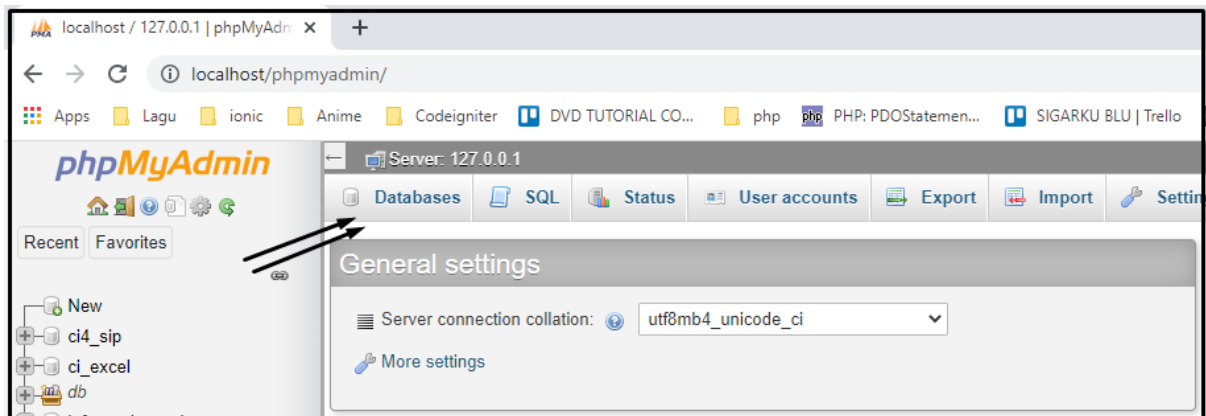
Perintah diatas digunakan untuk membuat sebuah tabel dengan nama **pegawai**, yang berisi kolom antara lain :

- id\_pegawai (Integer – Auto Increment – Primary Key)
  - nama (Varchar 255)
  - jenis\_kelamin (Enum['pria','wanita'])
  - no\_telp (Varchar 100)
  - email (Varchar 100)
  - alamat (Varchar 255)
  - created\_at (Date Time)
  - updated\_at (Date Time)
-

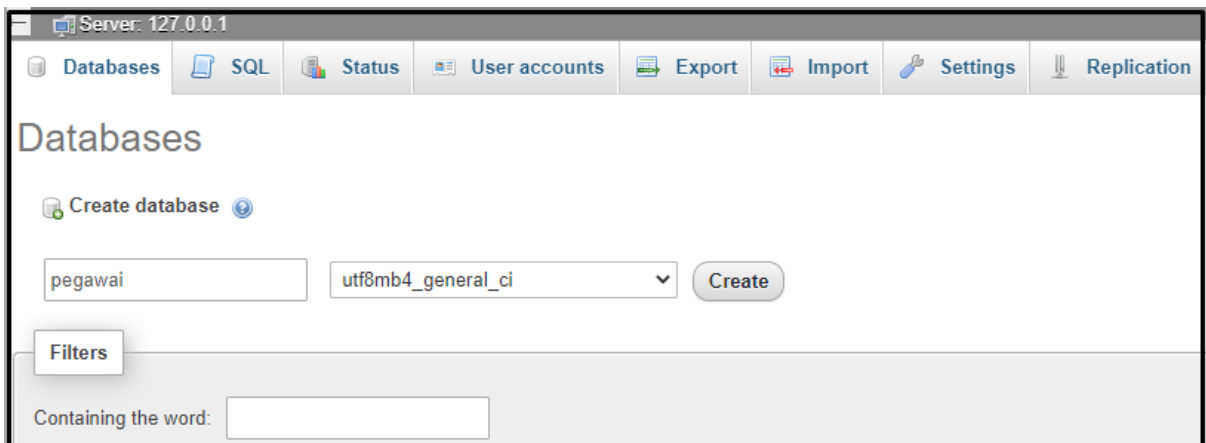
# Membuat Database

Setelah anda menuliskan perintah didalam migration untuk kebutuhan membuat tabel di database, berikutnya kita akan membuat database, didalam database ini akan kita buat tabel didalamnya dengan menggunakan perintah migration yang baru saja kita buat, langkah – langkahnya adalah sebagai berikut :

Silahkan akses alamat [localhost/phpmyadmin](http://localhost/phpmyadmin) di browser, tapi sebelum itu jangan lupa untuk mengaktifkan service [apache](#) dan [MySQL](#), lalu berikutnya silahkan klik menu Database untuk membuat database baru.



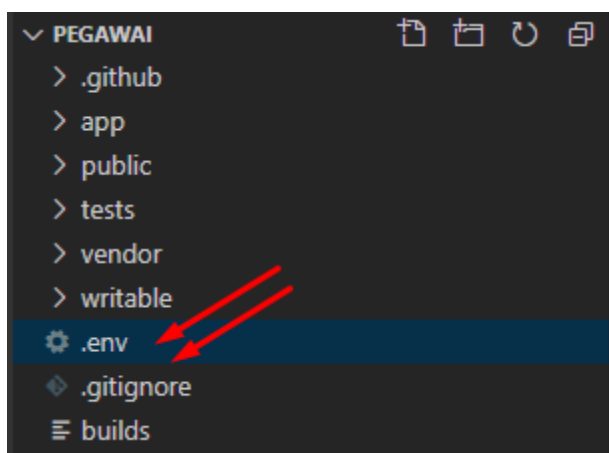
Berikutnya dibagian menu database silahkan [masukkan nama database](#) yang ingin dibuat., dalam contoh ini nama database adalah [pegawai](#), lalu kita klik tombol Create



# Setting Codeigniter 4 agar dapat terkoneksi dengan database

Setelah kita membuat database, dalam contoh ini adalah database **pagawai**, berikutnya kita akan setting codeigniter agar dapat terkoneksi dengan database, langkah – langkahnya adalah sebagai berikut :

didalam folder dari project codeigniter4 terdapat file dengan nama **env** silahkan **rename** menjadi **.env**



berikutnya buka file **.env** tersebut dan edit code didalamnya, untuk konfigurasi database berada pada line 52 – 56, hilangkan tanda # untuk mengaktifkan konfigurasi tersebut.

```
database.default.hostname = localhost  
database.default.database = pegawai  
database.default.username = root  
database.default.password =  
database.default.DBDriver = MySQLi
```

Keterangan :

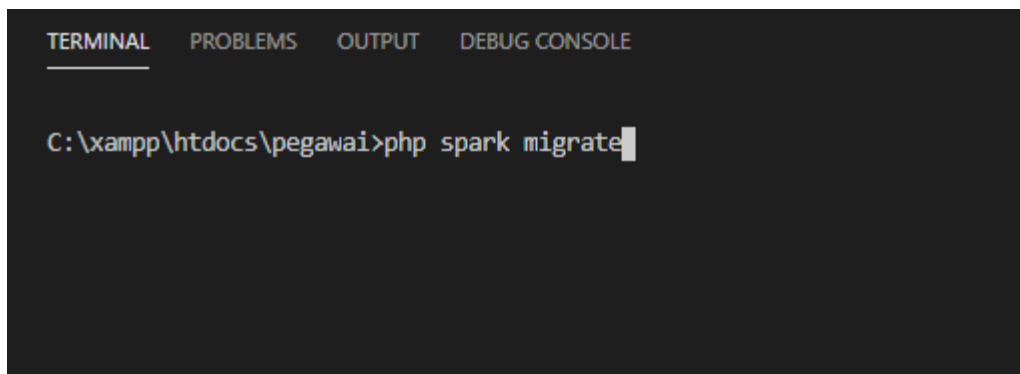
- **hostname** adalah alamat server dari database
- **database** adalah nama database yang barusan kita buat, jadi kita isi nilainya dengan nama pegawai
- **username** adalah nama user dari database dalam contoh ini adalah root
- **password** adalah password dari database dalam contoh ini saya biarkan kosong, karena default dari password user root di xampp itu kosong
- **DBDriver** saya biasanya default yaitu MySQLi

# Menjalankan File Migration untuk membuat tabel di database.

Berikutnya kita akan menjalankan file migration untuk membuat tabel di database, silahkan masuk ke project codeigniter anda melalui terminal, lalu jalankan perintah :

```
php spark migrate
```

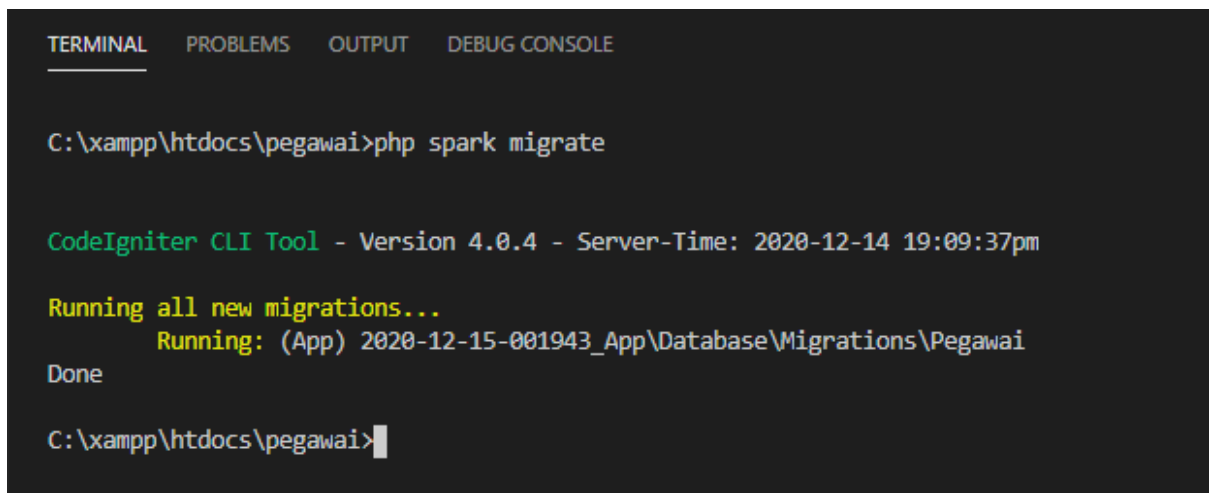
Jadi kurang lebih seperti ini tampilannya :



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

C:\xampp\htdocs\pegawai>php spark migrate
```

lalu tekan enter, jika proses berhasil maka tampilannya adalah sebagai berikut :



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

C:\xampp\htdocs\pegawai>php spark migrate

CodeIgniter CLI Tool - Version 4.0.4 - Server-Time: 2020-12-14 19:09:37pm

Running all new migrations...
Running: (App) 2020-12-15-001943_App\Database\Migrations\Pegawai
Done

C:\xampp\htdocs\pegawai>
```

maka sampai pada step ini kita sudah membuat tabel dengan nama **pegawai**, di database **pegawai** yang sebelumnya sudah kita buat.

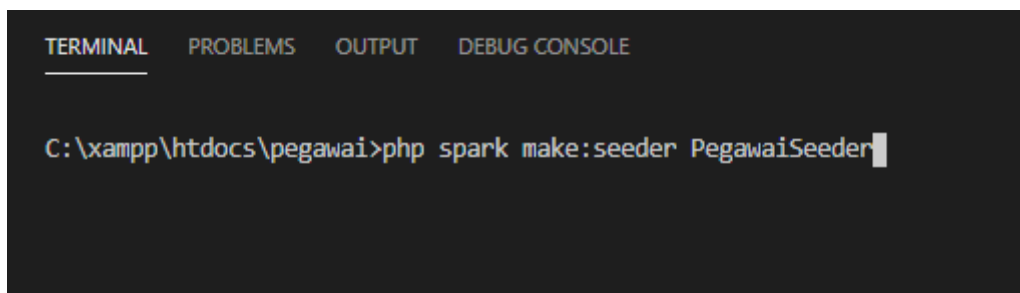
# Insert Dummy Data menggunakan Seeding

Berikutnya kita akan menambahkan **dummy data** didalam tabel pegawai menggunakan fitur **seeding**, sehingga nanti saat kita belajar untuk menampilkan data **pegawai** ke dalam bentuk tabel kita sudah memiliki beberapa data didalam tabel tersebut, langkah – langkahnya adalah sebagai berikut :

Silahkan masuk kedalam project **codeigniter 4** anda, melalui **terminal**, lalu jalankan perintah dibawah ini :

```
php spark make:seeder PegawaiSeeder
```

jadi kurang lebih seperti gambar dibawah ini :



lalu tekan enter, jika berhasil maka akan terbuat sebuah file dengan nama **PegawaiSeeder** didalam folder **app\Database\Seeds\**

buka file tersebut, lalu tuliskan code didalam file **PegawaiSeeder.php**

```
<?php
namespace App\Database\Seeds;

use CodeIgniter\Database\Seeder;
use CodeIgniter\I18n\Time;

class PegawaiSeeder extends Seeder
{
    public function run()
    {
        $data = [
            [
                'nama'      => 'Anton',
                'jenis_kelamin' => 'pria',
                'no_telp'    => '081234555678',
                'email'      => 'anton@gmail.com',
                'alamat'    => 'Jl. Mawar Putih No. 190, Waru Sidoarjo',
                'created_at' => Time::now()
            ]
        ],
```

```

[
    [
        'nama' => 'Budi',
        'jenis_kelamin' => 'pria',
        'no_telp' => '08571234567',
        'email' => 'budi@gmail.com',
        'alamat' => 'Jl. Melati Putih No. 77, Gedangan Sidoarjo',
        'created_at' => Time::now()
    ],
    [
        'nama' => 'Dita',
        'jenis_kelamin' => 'wanita',
        'no_telp' => '08122334455',
        'email' => 'dita@gmail.com',
        'alamat' => 'Jl. Rembulan No. 90, Krian Sidoarjo',
        'created_at' => Time::now()
    ]
];
$this->db->table('pegawai')->insertBatch($data);
}
}

```

Keterangan :

- dalam perintah diatas, kita menginsert 3 data pegawai dengan nama anton, budi, dan dita.

berikutnya kita akan jalankan perintah diatas untuk insert **data pegawai** tersebut kedalam tabel **pegawai**, silahkan masuk ke folder project **codeigniter** melalui terminal, lalu jalankan perintah sebagai berikut ini :

`php spark db:seed PegawaiSeeder`



The screenshot shows a Windows command prompt window with the following text:

```

Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\pegawai>php spark db:seed PegawaiSeeder

```

yang perlu teman – teman perhatikan adalah dibagian nama file yaitu PegawaiSeeder silahkan sesuaikan untuk bagian ini sesuaikan dengan nama file Seeder, lalu klik enter untuk menjalankan perintah tersebut

Lalu silahkan cek untuk tabel pegawai, harusnya sudah berisi 3 data pegawai yang kita insert melalui fitur seeder ini..



Server: 127.0.0.1 » Database: pegawai » Table: pegawai

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

`SELECT * FROM `pegawai``

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

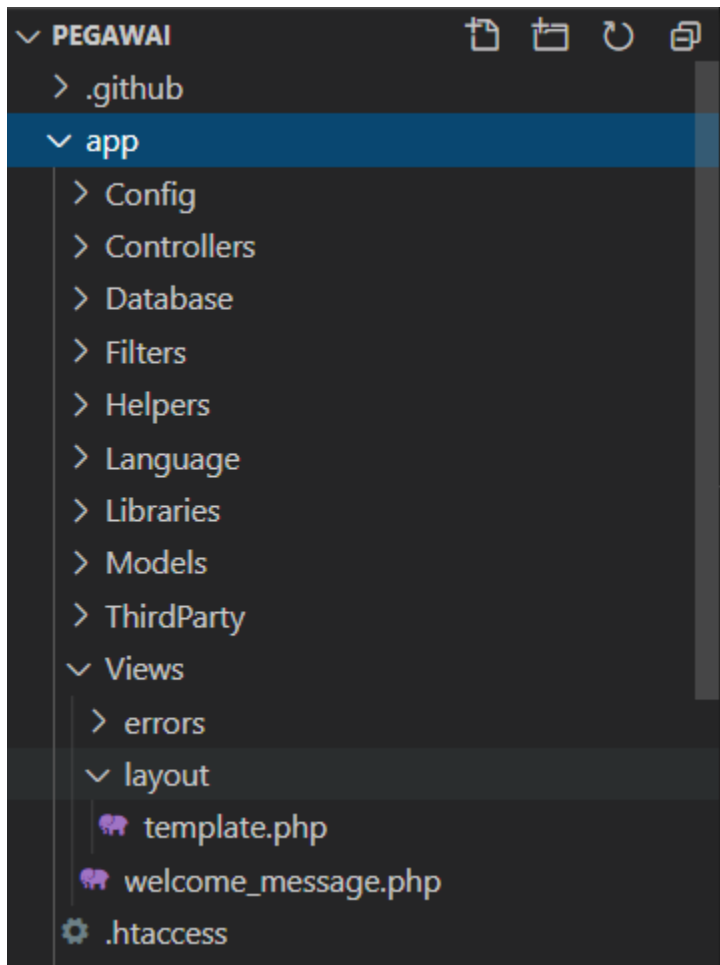
	id_pegawai	nama	jenis_kelamin	no_telp	email	alamat	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	Anton	pria	081234555678	anton@gmail.com	Jl. Mawar Putih No. 190, Waru Sidoarjo	2020-12-18 07:07:14	NULL
<input type="checkbox"/> Edit Copy Delete	2	Budi	pria	08571234567	budi@gmail.com	Jl. Melati Putih No. 77, Gedangan Sidoarjo	2020-12-18 07:07:14	NULL
<input type="checkbox"/> Edit Copy Delete	3	Dita	wanita	08122334455	dita@gmail.com	Jl. Rembulan No. 90, Krian Sidoarjo	2020-12-18 07:07:14	NULL

# Membuat Template untuk kebutuhan Tampilan

Berikutnya kita akan membuat template untuk kebutuhan tampilan, sehingga nanti ketika kita membuat bagian fitur untuk tampil data, tambah data, dan update data tampilannya akan sama, langkah – langkahnya adalah sebagai berikut :

Untuk langkah awal silahkan buat folder dengan nama **layout** didalam folder **app/Views**, lalu berikutnya buat file dengan nama **template.php** didalam folder layout tersebut, jadi kondisi saat ini ada sebuah file dengan nama **template.php** didalam folder **app/Views/layout**

jadi posisi filenya adalah seperti berikut ini :



lalu buka template.php dan isi dengan code sebagai berikut :

```
<!doctype html>
<html lang="en" class="h-100">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
  <meta name="generator" content="Jekyll v4.1.1">
  <title>Sticky Footer Navbar Template · Bootstrap</title>

  <link rel="canonical" href="https://getbootstrap.com/docs/4.5/examples/sticky-footer-navbar/">

  <!-- Bootstrap core CSS -->
  <link href="https://getbootstrap.com/docs/4.5/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0JlfiDPvq6uqKl2xXr2" crossorigin="anonymous">

  <style>
    .bd-placeholder-img {
      font-size: 1.125rem;
      text-align: middle;
      -webkit-user-select: none;
      -moz-user-select: none;
      -ms-user-select: none;
      user-select: none;
    }
  </style>
```

```

    }
    @media (min-width: 768px) {
        .bd-placeholder-img-lg {
            font-size: 3.5rem;
        }
    }
</style>
<!-- Custom styles for this template -->
<link href="https://getbootstrap.com/docs/4.5/examples/sticky-footer-navbar/sticky-footer-navbar.css"
rel="stylesheet">
</head>

<body class="d-flex flex-column h-100">
    <header>
        <!-- Fixed navbar -->
        <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
            <a class="navbar-brand" href="https://warungbelajar.com">Warung Belajar</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-
controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarCollapse">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="/home">Home <span class="sr-only">(current)</span></a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/pegawai">Pegawai</a>
                    </li>
                </ul>
            </div>
        </nav>
    </header>

    <!-- Begin page content -->
    <main role="main" class="flex-shrink-0">
        <?= $this->renderSection('content') ?>
    </main>

    <footer class="footer mt-auto py-3">
        <div class="container">
            <span class="text-muted">Place sticky footer content here.</span>
        </div>
    </footer>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://getbootstrap.com/docs/4.5/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"
crossorigin="anonymous"></script>
</body>

</html>

```

Keterangan :

- untuk tampilan ini saya ambil dari halaman example dari bootstrap, dengan link : <https://getbootstrap.com/docs/5.0/examples/sticky-footer-navbar/>
- untuk bagian **library css** dan **javascript** yang digunakan dibagian template, kita langsung akses secara online ya, untuk mempersingkat proses pembuatan template, jadi saya sarankan untuk menggunakan koneksi internet untuk

menjalankan hasilnya nanti., untuk **Library CSS** berada pada line (15 dan 34), Line (71 dan 72) untuk **Library JQuery**

Template ini memiliki 1 bagian yang akan digunakan sebagai bagian content, dimana nantinya akan berisi tampilan yang bersifat dinamis sesuai menu apa yang ditampilkan, untuk section tersebut kita beri nama content (Line 60)

## Membuat Model untuk komunikasi dengan tabel di database

Sebelum kita memulai proses membuat CRUD, kita persiapkan dulu **model** yang akan kita gunakan untuk berkomunikasi dengan tabel **pegawai** di database, untuk langkah awal buat file model dengan nama **PegawaiModel.php**, simpan file tersebut didalam folder **app/Models** lalu isi file **PegawaiModel.php** dengan code sebagai berikut :

```
<?php
namespace App\Models;

use CodeIgniter\Model;

class PegawaiModel extends Model
{
    protected $table = "pegawai";
    protected $primaryKey = "id_pegawai";
    protected $returnType = "object";
    protected $useTimestamps = true;
    protected $allowedFields = ['id_pegawai', 'nama', 'jenis_kelamin', 'no_telp', 'email', 'alamat'];
}
```

Keterangan :

- untuk classnya kita beri nama yang sama dengan nama filenya yaitu **PegawaiModel**
- beberapa settingan yang kita tulis adalah :
  - **\$table** dengan nilai pegawai karena kita akan komunikasi dengan tabel **pegawai**
  - **\$primaryKey** dengan nilai **id\_pegawai** karena kolom **id\_pegawai** merupakan primary key dari tabel **pegawai**
  - **\$returnType** dengan nilai **object** karena kita ingin untuk return type nanti berupa **object**
  - **\$useTimestamps** dengan nilai **true** karena kita akan mengisi kolom **created\_at** (saat insert data), dan kolom **updated\_at** (saat update data)
  - **\$allowedFields** kita isikan nama kolom di tabel pegawai yang boleh diinsert data..

setelah kita membuat model, berikutnya kita akan membuat menu untuk menampilkan data pegawai.

# Membuat Crud – Menampilkan Data Pegawai

Setelah di tahap persiapan, kita sudah mempersiapkan :

- Instalasi codeigniter 4
- Membuat tabel dengan migration
- Insert data ke tabel pegawai menggunakan seeder
- Membuat tampilan untuk kebutuhan template
- Membuat Model untuk komunikasi dengan tabel pegawai
- 

Berikutnya kita akan membuat menu untuk menampilkan data pegawai, langkah – langkahnya adalah sebagai berikut :

Buatlah sebuah controller dengan nama file **Pegawai.php** didalam folder **app/Controllers**, dan isi dengan code seperti berikut ini :

```
<?php

namespace App\Controllers;

use App\Models\PegawaiModel;

class Pegawai extends BaseController
{
    protected $pegawai;

    function __construct()
    {
        $this->pegawai = new PegawaiModel();
    }

    public function index()
    {
        $data['pegawai'] = $this->pegawai->findAll();
        return view('pegawai/index', $data);
    }
}
```

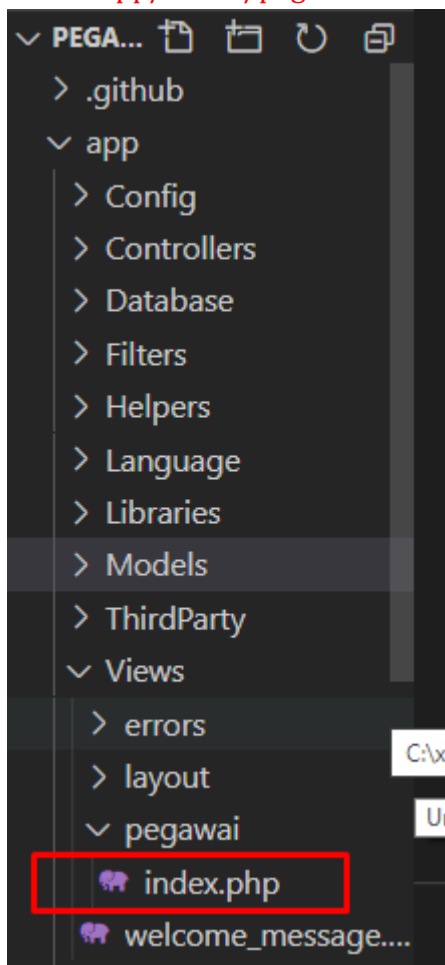
Keterangan :

- kita buat class dengan nama yang sama dengan nama filenya yaitu **Pegawai** (Line 7)
- kita load Model dengan **PegawaiModel** dengan menuliskan perintah **use App\Models\PegawaiModel;** (Line 5), kita load model ini agar bisa berkomunikasi dengan tabel **Pegawai**
- Kita buat variabel didalam class dengan nama **\$pegawai** (Line 9)
- kita buat method **\_\_construct** untuk membuat sebuah object **\$pegawai** dengan menggunakan class **PegawaiModel** (Line 11 – 14)

- Kita buat method **index** (Line 16) pada method **index** ini kita buat variabel **\$data** yang merupakan array.
- Berikutnya kita tuliskan perintah **\$this->pegawai->findAll()** yang digunakan untuk mengambil data pegawai di tabel **pegawai**, dan hasil data tersebut kita simpan ke dalam variabel **\$data** yang merupakan array dengan key **pegawai**. (Line 18).
- Berikutnya variabel **\$data** yang telah berisi array akan kita passing ke bagian **view** dengan nama **index** didalam folder **app/Views/pegawai** (Line 19)

## *Membuat View untuk menampilkan data pegawai*

Setelah kita membuat controller untuk mempersiapkan data pegawai yang akan ditampilkan dibagian view, berikutnya kita perlu membuat **view**, silahkan buat folder terlebih dahulu dengan nama **pegawai** didalam folder **app/Views**, kita buat folder agar **lebih mudah dalam management file view** untuk kebutuhan fitur **CRUD** data **pegawai** ini, setelah itu kita buat file dengan nama **index.php** didalam folder **app/Views/pegawai**



kurang lebih posisinya seperti gambar diatas, berikutnya kita isi file **index.php** dengan code sebagai berikut :

```
<?= $this->extend('layout/template'); ?>
<?= $this->section('content'); ?>
<div class="container">
    <div class="card">
        <div class="card-header">
            <h3>Data Pegawai</h3>
        </div>
        <div class="card-body">
            <?php if (!empty(session()->getFlashdata('message'))) : ?>
                <div class="alert alert-success alert-dismissible fade show" role="alert">
                    <?php echo session()->getFlashdata('message'); ?>
                    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
            <?php endif; ?>
            <a href="<?= base_url('/pegawai/create'); ?>" class="btn btn-primary">Tambah</a>
            <hr />
            <table class="table table-bordered">
                <tr>
                    <th>No</th>
                    <th>Nama</th>
                    <th>Jenis Kelamin</th>
                    <th>No Telp</th>
                    <th>Email</th>
                    <th>Alamat</th>
                    <th>Action</th>
                </tr>
                <?php
                $no = 1;
                foreach ($pegawai as $row) {
                    ?>
                    <tr>
                        <td><?= $no++; ?></td>
                        <td><?= $row->nama; ?></td>
                        <td><?= $row->jenis_kelamin; ?></td>
                        <td><?= $row->no_telp; ?></td>
                        <td><?= $row->email; ?></td>
                        <td><?= $row->alamat; ?></td>
                        <td>
                            <a title="Edit" href="<?= base_url('pegawai/edit/$row->id_pegawai'); ?>" class="btn btn-
info">Edit</a>
                            <a title="Delete" href="<?= base_url('pegawai/delete/$row->id_pegawai') ?>" class="btn btn-danger"
onclick="return confirm('Apakah Anda yakin ingin menghapus data ?')">Delete</a>
                        </td>
                    </tr>
                </tr>
                <?php
                }
                ?>
            </table>
        </div>
    </div>
</div>
<?= $this->endSection('content'); ?>
```

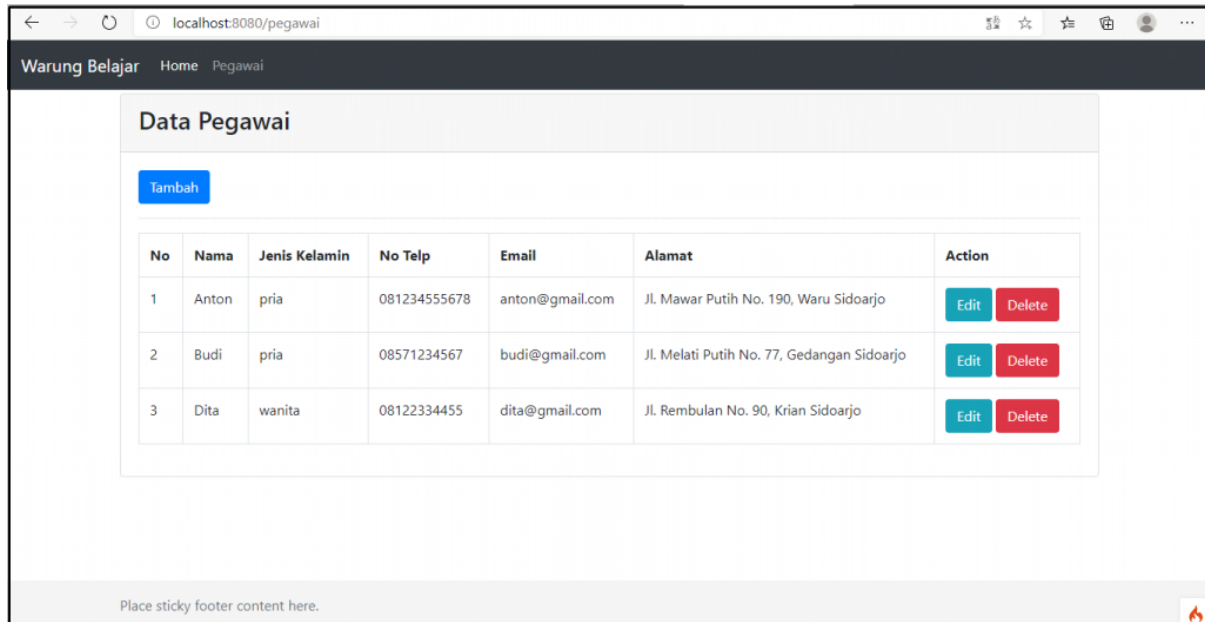
Berikutnya kita akan coba menjalankannya terlebih dahulu, kita jalankan dulu untuk **local development server**, silahkan ketikkan perintah :

```
php spark serve
```

lalu akses dengan url :

<http://localhost:8080/pegawai>

maka hasilnya harusnya seperti berikut ini :



kurang lebih tampilannya adalah seperti gambar diatas, akan ditampilkan 3 data pegawai yang sudah kita insert melalui fitur seed dibagian sebelumnya..

#### Keterangan Code di view index :

```
<?= $this->extend('layout/template'); ?>
<?= $this->section('content'); ?>
```

Code (Line 1 – 2) tersebut digunakan untuk **menentukan template yang digunakan**, dan kita menampilkan data pegawai didalam section **content** didalam layout yang sebelumnya kita telah buat..

```
<?php if (!empty(session()->getFlashdata('message'))) : ?>
    <div class="alert alert-success alert-dismissible fade show" role="alert">
        <?php echo session()->getFlashdata('message'); ?>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    </div>
<?php endif; ?>
```

Line 9 – 16 kita tuliskan perintah untuk melakukan pengecekan apakah ada **session flashdata** dengan nama **message**, jika ada maka akan ditampilkan data dari **session flashdata** tersebut, **session flashdata** dengan nama **message** ini akan terbuat ketika



proses **insert** dan **update** data berhasil, session ini akan **dikirimkan dari controller untuk ditampilkan dibagian view**

```
<a href="<?= base_url('/pegawai/create'); ?>" class="btn btn-primary">Tambah</a>
```

Code (Line 17) ini digunakan untuk menampilkan tombol tambah, yang ketika diklik akan mengakses URL **base\_url/pegawai/create**

Line 20 – 28 digunakan untuk membuat tabel header dari mulai kolom no, hingga action

```
foreach ($pegawai as $row) {  
    ?>  
    <tr>  
        <td><?= $no++; ?></td>  
        <td><?= $row->nama; ?></td>  
        <td><?= $row->jenis_kelamin; ?></td>  
        <td><?= $row->no_telp; ?></td>  
        <td><?= $row->email; ?></td>  
        <td><?= $row->alamat; ?></td>  
        <td>  
            <a title="Edit" href="<?= base_url("pegawai/edit/$row->id_pegawai"); ?>" class="btn btn-info">Edit</a>  
            <a title="Delete" href="<?= base_url("pegawai/delete/$row->id_pegawai") ?>" class="btn btn-danger" onclick="return confirm('Apakah Anda yakin ingin menghapus data ?')">Delete</a>  
        </td>  
    </tr>  
<?php  
}
```

Line 31 – 46 kita extract data array di variabel **\$pegawai** dengan menggunakan perintah **foreach**, dan setiap perulangannya menggunakan variabel **\$row**, dimana kita akan mengisi kolom disetiap baris datanya

```
<a title="Edit" href="<?= base_url("pegawai/edit/$row->id_pegawai"); ?>" class="btn btn-info">Edit</a>  
<a title="Delete" href="<?= base_url("pegawai/delete/$row->id_pegawai") ?>" class="btn btn-danger" onclick="return confirm('Apakah Anda yakin ingin menghapus data ?')">Delete</a>
```

Khusus untuk line 41 – 42 kita buat 2 tombol untuk kebutuhan edit dan delete, dimana untuk tombol edit akan mengakses url : **base\_url/pegawai/edit/id-pegawai yang diedit**, untuk tombol delete akan mengakses : **base\_url/pegawai/delete/id-pegawai yang didelete**

# Membuat Crud – Menambahkan Data Pegawai

Setelah dalam bagian sebelumnya kita telah menampilkan data pegawai, berikutnya kita akan membuat fitur untuk menambahkan data pegawai, langkah – langkahnya adalah sebagai berikut :

pada step awal kita bisa menambahkan routing untuk kebutuhan menampilkan form tambah data baru, silahkan buka file dengan nama **Routes.php** didalam folder **app/Config**

pada line 35 tambahkan code routing seperti berikut :

```
$routes->get('/pegawai', 'Pegawai::index');  
$routes->get('/pegawai/create', 'Pegawai::create');
```

Code diatas untuk kebutuhan routing dari fitur tampil data pegawai (Line 1), dan fitur untuk menampilkan form tambah data pegawai (Line 2)

Berikutnya kita buka controller **pegawai**, dan buat method dengan nama **create** , letakkan setelah method index , sehingga controller Pegawai menjadi seperti berikut ini :

```
<?php  
  
namespace App\Controllers;  
  
use App\Models\PegawaiModel;  
  
class Pegawai extends BaseController  
{  
    protected $pegawai;  
  
    function __construct()  
    {  
        $this->pegawai = new PegawaiModel();  
    }  
  
    public function index()  
    {  
        $data['pegawai'] = $this->pegawai->findAll();  
        return view('pegawai/index', $data);  
    }  
  
    public function create()  
    {
```

```

    return view('pegawai/create');
}
}

```

Perintah diatas digunakan saat method **create** ini diakses, maka akan menampilkan view dengan nama **create.php** didalam folder **app/Views/pegawai**

Berikutnya kita perlu membuat file view dengan **create.php** didalam folder **app/Views/pegawai**, untuk code di file **create.php** adalah sebagai berikut :

```

<?= $this->extend('layout/template'); ?>
<?= $this->section('content'); ?>
<div class="container">
    <div class="card">
        <div class="card-header">
            <h3>Tambah Data Pegawai</h3>
        </div>
        <div class="card-body">
            <?php if (!empty(session()->getFlashdata('error'))): ?>
                <div class="alert alert-danger alert-dismissible fade show" role="alert">
                    <h4>Periksa Entrian Form</h4>
                    </hr />
                    <?php echo session()->getFlashdata('error'); ?>
                    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
            <?php endif; ?>
            <form method="post" action="<?= base_url('pegawai/store') ?>">
                <?= csrf_field(); ?>

                <div class="form-group">
                    <label for="nama">Nama</label>
                    <input type="text" class="form-control" id="nama" name="nama" value="<?= old('nama'); ?>">
                </div>

                <div class="form-group">
                    <label for="jenis_kelamin">Jenis Kelamin</label>
                    <select name="jenis_kelamin" class="form-control" id="jenis_kelamin">
                        <option value="pria">Pria</option>
                        <option value="wanita">Wanita</option>
                    </select>
                </div>

                <div class="form-group">
                    <label for="no_telp">No Telp</label>
                    <input type="text" class="form-control" id="no_telp" name="no_telp" value="<?= old('no_telp') ?>" />
                </div>

                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="text" class="form-control" id="email" name="email" value="<?= old('email') ?>" />
                </div>

                <div class="form-group">
                    <label for="alamat">Alamat</label>
                    <textarea class="form-control" name="alamat" id="alamat"><?= old('alamat') ?></textarea>
                </div>

                <div class="form-group">
                    <input type="submit" value="Simpan" class="btn btn-info" />
                </div>
            </form>
        </div>
    </div>
</div>

```

```

        </div>
    </form>
</div>
</div>
</div>
<?= $this->endSection('content'); ?>

```

Kita coba mengakses form tersebut terlebih dahulu, jangan lupa menjalankan local development server dengan perintah terminal :

```
php spark serve
```

lalu akses dengan URL :

```
localhost:8080/pegawai/create
```

hasilnya adalah seperti berikut ini :

Tampilannya kurang lebih seperti gambar diatas, berikutnya saya akan bahas code yang saya tulis :

```

<?= $this->extend('layout/template'); ?>
<?= $this->section('content'); ?>

```

Code (Line 1 – 2) tersebut digunakan untuk menentukan **template** yang digunakan, dan kita menampilkan data pegawai didalam section **content** didalam layout yang sebelumnya kita telah buat..

```

<?php if (!empty(session()->getFlashdata('error'))): ?>
    <div class="alert alert-danger alert-dismissible fade show" role="alert">
        <h4>Periksa Entrian Form</h4>
    </hr />

```

```

<?php echo session()->getFlashdata('error'); ?>
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
  <span aria-hidden="true">&times;</span>
</button>
</div>
<?php endif; ?>

```

Line 9 – 18 kita gunakan untuk menampilkan data error yang kita kirim melalui **session flashdata**, session flashdata ini kita kirim melalui **controller Pegawai**, session flashdata ini bernama error yang akan dikirimkan saat **form validasi di controller bernilai false**, dan ketika **error** itu terjadi maka akan menampilkan **pesan error** yang terjadi dari inputan form yang tidak sesuai dengan aturan form validasi

```

<form method="post" action="<?= base_url('pegawai/store') ?>">

```

Kita gunakan tag form dengan method **post**, dibagian attribute **action** akan mengakses url **base\_url/pegawai/store**, kita akan buat routenya nanti ketika url tersebut diakses akan diarahkan ke method **store** di controller **pegawai**, sehingga ketika form ini disubmit, maka akan di proses di method **store** pada controller **pegawai**.

```

<?= csrf_field(); ?>

```

ini untuk kebutuhan membuat input bertipekan **hidden**, untuk kebutuhan **token csrf**

```

<div class="form-group">
  <label for="nama">Nama</label>
  <input type="text" class="form-control" id="nama" name="nama" value="<?= old('nama'); ?>">
</div>

<div class="form-group">
  <label for="jenis_kelamin">Jenis Kelamin</label>
  <select name="jenis_kelamin" class="form-control" id="jenis_kelamin">
    <option value="pria">Pria</option>
    <option value="wanita">Wanita</option>
  </select>
</div>

<div class="form-group">
  <label for="no_telp">No Telp</label>
  <input type="text" class="form-control" id="no_telp" name="no_telp" value="<?= old('no_telp') ?>" />
</div>

<div class="form-group">
  <label for="email">Email</label>
  <input type="text" class="form-control" id="email" name="email" value="<?= old('email') ?>" />
</div>

<div class="form-group">
  <label for="alamat">Alamat</label>
  <textarea class="form-control" name="alamat" id="alamat"><?= old('alamat') ?></textarea>
</div>

<div class="form-group">
  <input type="submit" value="Simpan" class="btn btn-info" />
</div>

```

berikutnya kita tulis code HTML untuk kebutuhan setiap bagian inputan, yang terpenting setiap inputan form memiliki **value** di **attribute name**, dan untuk **attribute value** disini kita gunakan **method old** yang artinya akan menampilkan inputan form sebelumnya saat error form validasi terjadi.

## *Membuat Proses penyimpanan data pegawai*

Setelah dibagian sebelumnya kita sudah membuat form untuk proses tambah data, berikutnya kita akan buat untuk method yang digunakan menyimpan data pegawai, tetapi sebelum itu kita buat routing terlebih dahulu, silahkan buka file dengan nama **app/Config/Routes.php**, tambah routing seperti berikut ini :

```
$routes->post('/pegawai/store', 'Pegawai::store');
```

Berikutnya kita buka controller **Pegawai.php**, lalu tambahkan method **store**, sehingga code controller **Pegawai** menjadi seperti berikut ini :

```
<?php

namespace App\Controllers;

use App\Models\PegawaiModel;

class Pegawai extends BaseController
{
    protected $pegawai;

    function __construct()
    {
        $this->pegawai = new PegawaiModel();
    }

    public function index()
    {
        $data['pegawai'] = $this->pegawai->findAll();
        return view('pegawai/index', $data);
    }

    public function create()
    {
        return view('pegawai/create');
    }

    public function store()
    {
        if (!$this->validate([
            'nama' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'jenis_kelamin' => [
                'rules' => 'required',
                'errors' => [

```

```

        'required' => '{field} Harus diisi'
    ],
    'no_telp' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],
    'email' => [
        'rules' => 'required|valid_email',
        'errors' => [
            'required' => '{field} Harus diisi',
            'valid_email' => 'Email Harus Valid'
        ]
    ],
    'alamat' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],
],
))) {
    session()->setFlashdata('error', $this->validator->listErrors());
    return redirect()->back()->withInput();
}

$this->pegawai->insert([
    'nama' => $this->request->getVar('nama'),
    'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
    'no_telp' => $this->request->getVar('no_telp'),
    'email' => $this->request->getVar('email'),
    'alamat' => $this->request->getVar('alamat')
]);
session()->setFlashdata('message', 'Tambah Data Pegawai Berhasil');
return redirect()->to('/pegawai');
}
}

```

Keterangan :

Code untuk kebutuhan method **store** ada di line **27 – 76**

```

if (!$this->validate([
    'nama' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],
    'jenis_kelamin' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],
    'no_telp' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ]

```

```

    ],
    'email' => [
        'rules' => 'required|valid_email',
        'errors' => [
            'required' => '{field} Harus diisi',
            'valid_email' => 'Email Harus Valid'
        ]
    ],
    'alamat' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],
],
))) {

```

Line 29 – 62 kita menuliskan aturan (**rules**) form **validasi**, dengan aturan sebagai berikut :

- inputan **nama** harus diisi (Line 31)
- inputan **jenis\_kelamin** harus diisi (Line 37)
- inputan **no\_telp** harus diisi (Line 43)
- inputan **email** harus diisi, dan format **email harus valid** (Line 49)
- inputan **alamat** harus diisi (Line 56)

```

session()->setFlashdata('error', $this->validator->listErrors());
return redirect()->back()->withInput();

```

Line 63 – 64 code ini yang akan dijalankan **saat validasi inputan form ada yang tidak sesuai dengan rules di form validation**, kita akan **redirect ke halaman sebelumnya** (Form tambah Pegawai) dengan **menyertakan inputan di form** sebelumnya, selain itu juga membuat **session flasdata** dengan nama **error** yang akan berisi **list data error**.

data error dari session flashdata ini akan dicek di **form tambah pegawai**, anda bisa perhatikan di file **app/Views/pegawai/create.php** pada line 9 – 18

```

$this->pegawai->insert([
    'nama' => $this->request->getVar('nama'),
    'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
    'no_telp' => $this->request->getVar('no_telp'),
    'email' => $this->request->getVar('email'),
    'alamat' => $this->request->getVar('alamat')
]);

```

Line 67 – 73 kita melakukan perintah **insert data ke tabel pegawai**, kita gunakan **object pegawai** dan menggunakan method **insert**.



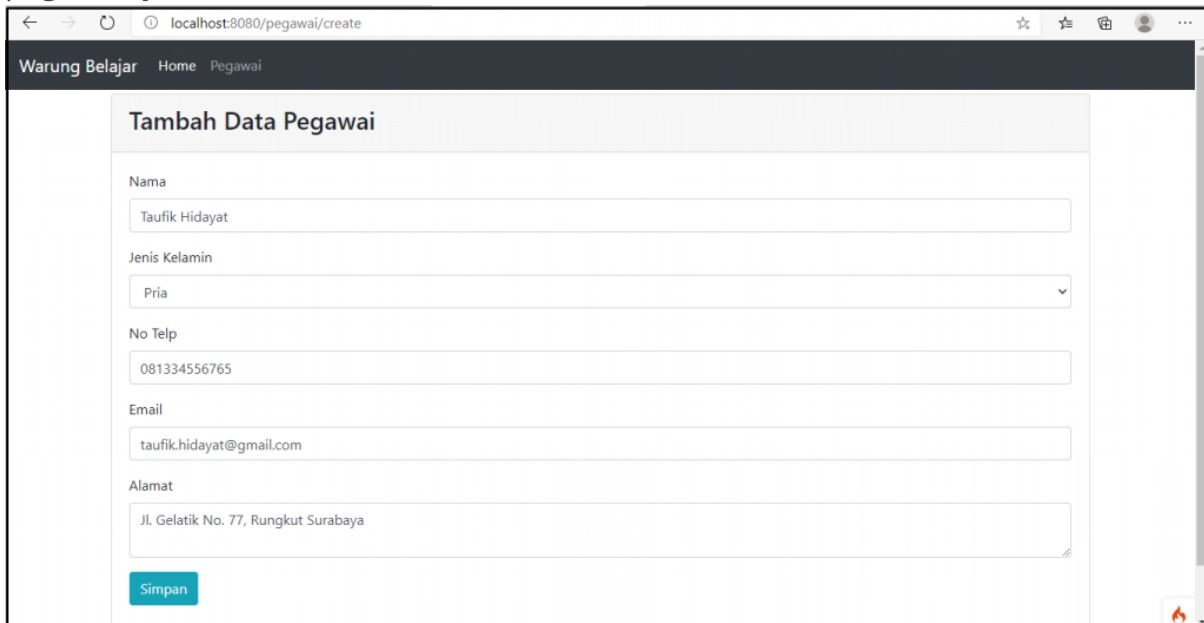
Pada method insert kita kirimkan parameter yang berisi array, dengan key disesuaikan dengan nama kolom di tabel pegawai dan value berisi inputan dari form, kita menggunakan perintah `$this->request->getVar('nama inputan form')` untuk menangkap inputan formnya.

```
session()->setFlashdata('message', 'Tambah Data Pegawai Berhasil');  
return redirect()->to('/pegawai');
```

Line 74 – 75 perintah ini akan dijalankan ketika proses insert data ke tabel pegawai selesai, yang dilakukan adalah redirect ke url `base_url/pegawai`, dan membuat session flashdata dengan nama message yang berisi text “Tambah Data Pegawai Berhasil”, pesan ini akan ditampilkan pada menu tampil data pegawai dibagian file view yang berada di `app/views/pegawai/index.php`

## Testing Proses Insert data

Silahkan akses alamat : `localhost:8080/pegawai/create`, jangan lupa menjalankan local development server terlebih dahulu, setelah tampil form nya silahkan isi data pegawainya, lalu tekan tombol submit.

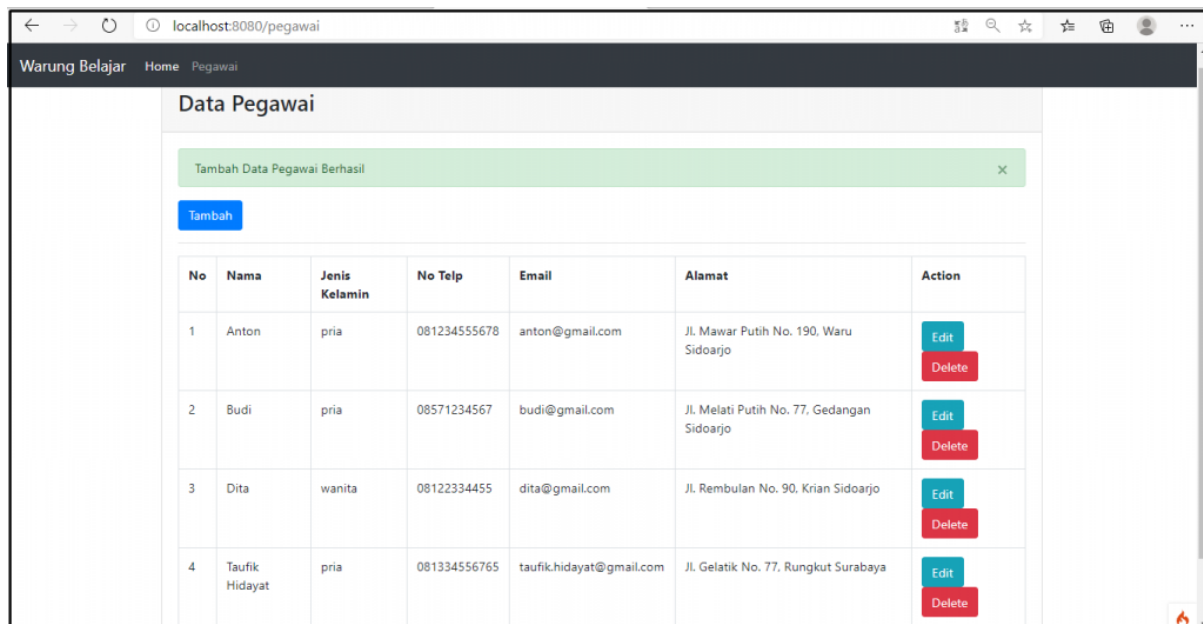


The screenshot shows a web browser window with the address bar displaying `localhost:8080/pegawai/create`. The page has a dark navigation bar with the text "Warung Belajar" and links for "Home" and "Pegawai". The main content area is titled "Tambah Data Pegawai" and contains a form with the following fields:

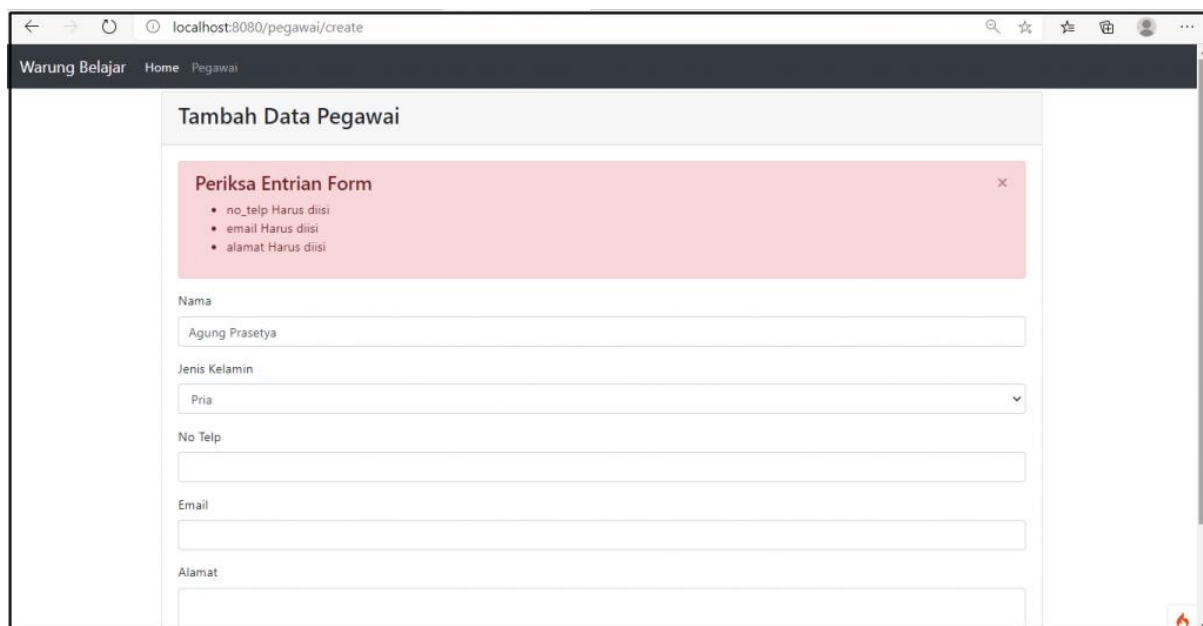
- Nama:** Taufik Hidayat
- Jenis Kelamin:** Pria (selected from a dropdown menu)
- No Telp:** 081334556765
- Email:** taufik.hidayat@gmail.com
- Alamat:** Jl. Gelatik No. 77, Rungkut Surabaya

At the bottom of the form is a blue button labeled "Simpan".

Hasilnya adalah seperti ini jika proses insert data berhasil, akan terlihat data baru, dan ada notifikasi bahwa “Tambah Data Pegawai Berhasil”



Ini tampilan jika terjadi error di bagian form validasi



## Membuat Crud – Update Data Pegawai

Setelah dalam tutorial sebelumnya kita telah belajar mengenai bagaimana cara menampilkan data pegawai, dan menambahkan data pegawai, berikutnya kita akan membuat fitur yang digunakan untuk **update data pegawai**, untuk proses update data pegawai ini akan kita bagi 2 bagian antara lain :

- Menampilkan form edit data pegawai, yang berisi data pegawai yang akan diedit
- Membuat method untuk proses update data pegawai

## Membuat Form edit pegawai

Pada saat kita membuat menu untuk menampilkan data pegawai, pada file view yang berada di folder `app/views/pegawai/index.php`, perhatikan pada line 41 kita buat tombol untuk kebutuhan edit data pegawai

```
<a title="Edit" href="<?= base_url("pegawai/edit/$row->id_pegawai"); ?>" class="btn btn-info">Edit</a>
```

jika kita melihat code untuk link diatas, ketika tombol edit di klik maka akan mengakses url : `base_url/pegawai/edit/data-id-pegawai`

Langkah awal kita buat routing terlebih dahulu, silahkan buka file `Routes.php` di folder `app/Config/Routes.php` tambahkan routing sebagai berikut :

```
$routes->get('pegawai/edit/(:num)', 'Pegawai::edit/$1');
```

kita buat routing dengan menggunakan fitur placeholder, kita membatasi untuk **parameter yang dikirimkan adalah berupa angka**, dan routing ini akan di proses di **Controller Pegawai** pada method **edit**,

Setelah proses pembuatan routing, berikutnya kita akan membuat method **edit** di controller **Pegawai**, silahkan buka controller `Pegawai.php`, dan buat method **edit** sehingga code yang ada di controller `Pegawai.php` adalah sebagai berikut ini :

```
<?php

namespace App\Controllers;

use App\Models\PegawaiModel;

class Pegawai extends BaseController
{
    protected $pegawai;

    function __construct()
    {
        $this->pegawai = new PegawaiModel();
    }

    public function index()
    {
        $data['pegawai'] = $this->pegawai->findAll();
        return view('pegawai/index', $data);
    }

    public function create()
    {
        return view('pegawai/create');
    }
}
```

```

public function store()
{
    if (!$this->validate([
        'nama' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'jenis_kelamin' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'no_telp' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'email' => [
            'rules' => 'required|valid_email',
            'errors' => [
                'required' => '{field} Harus diisi',
                'valid_email' => 'Email Harus Valid'
            ]
        ],
        'alamat' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ]
    ])) {
        session()->setFlashdata('error', $this->validator->listErrors());
        return redirect()->back()->withInput();
    }

    $this->pegawai->insert([
        'nama' => $this->request->getVar('nama'),
        'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
        'no_telp' => $this->request->getVar('no_telp'),
        'email' => $this->request->getVar('email'),
        'alamat' => $this->request->getVar('alamat')
    ]);
    session()->setFlashdata('message', 'Tambah Data Pegawai Berhasil');
    return redirect()->to('/pegawai');
}

function edit($id)
{
    $dataPegawai = $this->pegawai->find($id);
    if (empty($dataPegawai)) {
        throw new \CodeIgniter\Exceptions\PageNotFoundException('Data Pegawai Tidak ditemukan !');
    }
    $data['pegawai'] = $dataPegawai;
    return view('pegawai/edit', $data);
}
}

```

Keterangan :

Code method edit berada pada line 78 – 86.

pada method edit ini memiliki 1 parameter yaitu \$id, dimana \$id akan berisi id data pegawai yang akan di edit

```
$dataPegawai = $this->pegawai->find($id);
```

Line 80 kita menuliskan perintah untuk mendapatkan **data pegawai** yang nilai pada kolom **id\_pegawai** sesuai dengan nilai pada variabel **\$id**, karena kita mencari **data pegawai** berdasarkan kolom yang merupakan **primary key** sehingga bisa menggunakan method **find** dengan menggunakan **object pegawai**, hasil data pegawai ini akan kita simpan pada variabel **\$dataPegawai**

```
if (empty($dataPegawai)) {  
    throw new \CodeIgniter\Exceptions\PageNotFoundException('Data Pegawai Tidak ditemukan !');  
}
```

Code ini digunakan untuk melakukan **pengecekan apakah ada data pegawai hasil dari pencarian pegawai berdasarkan kolom primary key**, jika tidak ada maka akan memunculkan pesan **"Data Pegawai Tidak ditemukan"**

```
$data['pegawai'] = $dataPegawai;  
return view('pegawai/edit', $data);
```

Setelah itu data pegawai yang telah ditemukan akan disimpan pada variabel **\$data** yang merupakan **array**, kita simpan pada **array \$data** dengan **key pegawai**, dan kita load **view** dengan nama **edit** didalam folder **app/views/pegawai** dan kita **passing variabel \$data** yang berisi data **pegawai**, pada form edit nanti kita bisa menampilkan data pegawai yang akan di edit

Berikutnya kita akan membuat file view dengan nama **edit.php** didalam folder **app/Views/pegawai**, untuk code didalam file tersebut adalah sebagai berikut :

```
<?= $this->extend('layout/template'); ?>  
<?= $this->section('content'); ?>  
<div class="container">  
    <div class="card">  
        <div class="card-header">  
            <h3>Update Data Pegawai</h3>  
        </div>  
        <div class="card-body">  
            <?php if (!empty(session()->getFlashdata('error'))) : ?>  
                <div class="alert alert-danger alert-dismissible fade show" role="alert">  
                    <h4>Periksa Entrian Form</h4>  
                    </div>  
            <?php echo session()->getFlashdata('error'); ?>  
            <button type="button" class="close" data-dismiss="alert" aria-label="Close">  
                <span aria-hidden="true">&times;</span>  
            </button>  
        </div>  
        <?php endif; ?>  
        <form method="post" action="<?= base_url('pegawai/update/' . $pegawai->id_pegawai) ?>">  
            <?= csrf_field(); ?>
```

```

<div class="form-group">
  <label for="nama">Nama</label>
  <input type="text" class="form-control" id="nama" name="nama" value="<?= $pegawai->nama; ?>" />
</div>

<div class="form-group">
  <label for="jenis_kelamin">Jenis Kelamin</label>
  <select name="jenis_kelamin" class="form-control" id="jenis_kelamin">
    <option value="pria" <?= ($pegawai->jenis_kelamin == "pria" ? "selected" : ""); ?>>Pria</option>
    <option value="wanita" <?= ($pegawai->jenis_kelamin == "wanita" ? "selected" : ""); ?>>Wanita</option>
  </select>
</div>

<div class="form-group">
  <label for="no_telp">No Telp</label>
  <input type="text" class="form-control" id="no_telp" name="no_telp" value="<?= $pegawai->no_telp; ?>" />
</div>

<div class="form-group">
  <label for="email">Email</label>
  <input type="text" class="form-control" id="email" name="email" value="<?= $pegawai->email; ?>" />
</div>
<div class="form-group">
  <label for="alamat">Alamat</label>
  <textarea class="form-control" name="alamat" id="alamat"><?= $pegawai->alamat; ?></textarea>
</div>

<div class="form-group">
  <input type="submit" value="Update" class="btn btn-info" />
</div>

</form>
</div>
</div>
<?= $this->endSection('content'); ?>

```

Keterangan :

```

<?= $this->extend('layout/template'); ?>
<?= $this->section('content'); ?>

```

Line 1 – 2 digunakan untuk menset template mana yang digunakan dan untuk tampilannya diletakkan di bagian section **content** di template

```

<?php if (!empty(session()->getFlashdata('error'))) : ?>
  <div class="alert alert-danger alert-dismissible fade show" role="alert">
    <h4>Periksa Entrian Form</h4>
    </hr />
    <?php echo session()->getFlashdata('error'); ?>
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
<?php endif; ?>

```

Line 9 – 18 kita gunakan untuk menampilkan data error yang kita kirim melalui `session flashdata`, `session flashdata` ini kita kirim melalui controller `Pegawai`, `session flashdata` ini bernama `error` yang akan dikirimkan saat form validasi di controller pada method edit ada yang tidak sesuai rules yang dibuat, dan ketika `error` itu terjadi maka akan menampilkan pesan `error` dan ditampilkan di form edit

```
<form method="post" action="<?= base_url('pegawai/update/' . $pegawai->id_pegawai) ?>">
```

Kita menggunakan tag form HTML dengan method `Post`, dengan attribute action bernilai : `<?= base_url('pegawai/update/' . $pegawai->id_pegawai) ?>` yang artinya ketika tombol submit di form update ini di klik maka akan mengakses url: `base_url/pegawai/update/data-id-pegawai`, nantinya kita akan buat routing untuk handling url ini agar mengakses method update di controller `Pegawai`

```
<div class="form-group">
  <label for="nama">Nama</label>
  <input type="text" class="form-control" id="nama" name="nama" value="<?= $pegawai->nama; ?>">
</div>
```

Menampilkan form input `nama` dan memiliki value dari data `nama pegawai yang di edit`, hal tersebut karena kita menset nilai dari attribute `value` dengan perintah `$pegawai->nama`, yang artinya menampilkan nilai dari kolom `nama` di tabel `pegawai`, hal ini juga berlaku untuk inputan `jenis_kelamin`, `no_telp`, `email`, dan `alamat`

## *Membuat Method Update untuk memproses update data Pegawai*

Setelah kita membuat form untuk kebutuhan `edit pegawai`, berikutnya kita perlu membuat `method update` di controller `Pegawai` yang digunakan untuk memproses `update data pegawai`, tetapi sebelumnya itu kita perlu menambahkan `routing` untuk kebutuhan proses `update data` ini, silahkan buka file `Routes.php` di folder `app/Config/Routes.php`, tambahkan code routing seperti berikut ini :

```
$routes->post('pegawai/update/(:num)', 'Pegawai::update/$1');
```

Routing tersebut digunakan untuk handling untuk proses `update data`, jadi ketika `submit form update`, akan mengarah ke URL : `base_url/pegawai/update/id-`

pegawai, dan karena ada routing diatas akan diarahkan ke Controller Pegawai dengan method update

Berikutnya kita akan buat method update didalam controller Pegawai, silahkan buka Controller Pegawai lalu tambahkan method update, sehingga codenya menjadi seperti berikut ini :

```
<?php
namespace App\Controllers;

use App\Models\PegawaiModel;

class Pegawai extends BaseController
{
    protected $pegawai;

    function __construct()
    {
        $this->pegawai = new PegawaiModel();
    }

    public function index()
    {
        $data['pegawai'] = $this->pegawai->findAll();
        return view('pegawai/index', $data);
    }

    public function create()
    {
        return view('pegawai/create');
    }

    public function store()
    {
        if (!$this->validate([
            'nama' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'jenis_kelamin' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'no_telp' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'email' => [
                'rules' => 'required|valid_email',
                'errors' => [
                    'required' => '{field} Harus diisi',
                    'valid_email' => 'Email Harus Valid'
                ]
            ],
            'alamat' => [
```



```

        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],

    )) {
        session()->setFlashdata('error', $this->validator->listErrors());
        return redirect()->back()->withInput();
    }

    $this->pegawai->insert([
        'nama' => $this->request->getVar('nama'),
        'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
        'no_telp' => $this->request->getVar('no_telp'),
        'email' => $this->request->getVar('email'),
        'alamat' => $this->request->getVar('alamat')
    ]);
    session()->setFlashdata('message', 'Tambah Data Pegawai Berhasil');
    return redirect()->to('/pegawai');
}

function edit($id)
{
    $dataPegawai = $this->pegawai->find($id);
    if (empty($dataPegawai)) {
        throw new \CodeIgniter\Exceptions\PageNotFoundException('Data Pegawai Tidak ditemukan !');
    }
    $data['pegawai'] = $dataPegawai;
    return view('pegawai/edit', $data);
}

public function update($id)
{
    if (!$this->validate([
        'nama' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'jenis_kelamin' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'no_telp' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'email' => [
            'rules' => 'required|valid_email',
            'errors' => [
                'required' => '{field} Harus diisi',
                'valid_email' => 'Email Harus Valid'
            ]
        ],
        'alamat' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ]
    ]),

```

```

    )) {
        session()->setFlashdata('error', $this->validator->listErrors());
        return redirect()->back();
    }

    $this->pegawai->update($id, [
        'nama' => $this->request->getVar('nama'),
        'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
        'no_telp' => $this->request->getVar('no_telp'),
        'email' => $this->request->getVar('email'),
        'alamat' => $this->request->getVar('alamat')
    ]);
    session()->setFlashdata('message', 'Update Data Pegawai Berhasil');
    return redirect()->to('/pegawai');
}
}

```

Keterangan :

Method Update berada pada Line 88 – 137

Line 90 – 123 adalah code untuk kebutuhan **form validasi**, pada validasi ini kita buat beberapa rules antara lain :

- inputan **nama** harus diisi (Line 92)
- inputan **jenis\_kelamin** harus diisi (Line 98)
- inputan **no\_telp** harus diisi (Line 104)
- inputan **email** harus diisi, dan format **email** harus valid (Line 110)
- inputan **alamat** harus diisi (117)
- `session()->setFlashdata('error', $this->validator->listErrors());`
- `return redirect()->back();`

Line 124 – 125 akan dijalankan ketika ada error dibagian form validasi, akan membuat **session flashdata** dengan nama **error** yang berisi daftar **errornya**, dan kan redirect ke tampilan sebelumnya, dalam hal ini adalah form edit.

```

$this->pegawai->update($id, [
    'nama' => $this->request->getVar('nama'),
    'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
    'no_telp' => $this->request->getVar('no_telp'),
    'email' => $this->request->getVar('email'),
    'alamat' => $this->request->getVar('alamat')
]);

```

Perintah Line 128 – 134 digunakan untuk proses **update data pegawai**, kita gunakan object **\$this->pegawai**, dan kita gunakan method **update** untuk proses **update data** ke tabel **pegawai**

pada method **update**, terdapat **2 parameter** yaitu :

- **\$id** merupakan **id** data yang ingin **diupdate**, dalam hal ini adalah **id\_pegawai** dari **data pegawai yang akan diupdate**
- **parameter kedua** adalah **data yang diupdate**, kita bisa menggunakan **data array**, dimana untuk **key** kita gunakan **nama kolom** di tabel **pegawai**, dan untuk **value** adalah **hasil inputan dari form edit** kita gunakan perintah **\$this->request->getVar("");**

■  
Terkait penjelasan penggunaan method **update**, anda bisa membaca tutorial kita sebelumnya : [Membuat Model di Codeigniter 4](#) disana dibahas beberapa method yang bisa digunakan pada model, salah satunya adalah method update.

---

## *Testing Fitur Update data di data Pegawai*

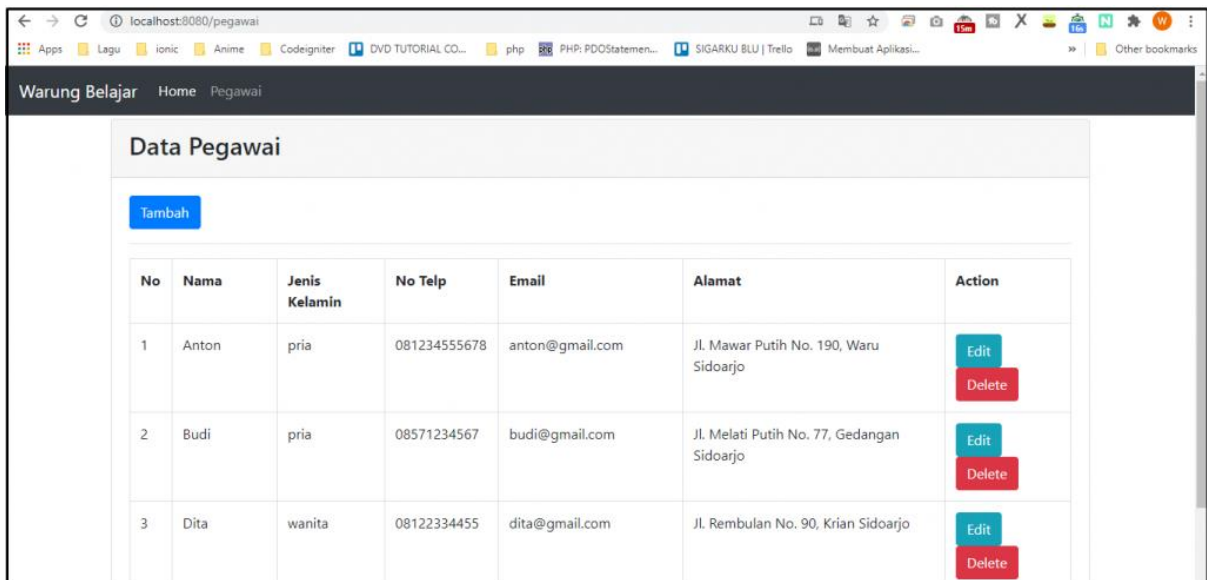
Setelah kita belajar bagaimana cara membuat fitur update data, berikutnya kita akan coba untuk melakukan update data pegawai, silahkan buka menu tampil data pegawai, jangan lupa menjalankan local development server terlebih dahulu dengan perintah :

`php spark serve`

lalu akses urlnya :

<http://localhost:8080/pegawai>

Maka akan menampilkan daftar pegawai :



Data Pegawai						
<a href="#">Tambah</a>						
No	Nama	Jenis Kelamin	No Telp	Email	Alamat	Action
1	Anton	pria	081234555678	anton@gmail.com	Jl. Mawar Putih No. 190, Waru Sidoarjo	<a href="#">Edit</a> <a href="#">Delete</a>
2	Budi	pria	08571234567	budi@gmail.com	Jl. Melati Putih No. 77, Gedangan Sidoarjo	<a href="#">Edit</a> <a href="#">Delete</a>
3	Dita	wanita	08122334455	dita@gmail.com	Jl. Rembulan No. 90, Krian Sidoarjo	<a href="#">Edit</a> <a href="#">Delete</a>

semisal kita ingin edit data pegawai dengan nama Anton, maka anda bisa klik tombol Edit..

localhost:8080/pegawai/edit/1

Warung Belajar Home Pegawai

### Update Data Pegawai

Nama  
Anton

Jenis Kelamin  
Pria

No Telp  
081234555678

Email  
anton@gmail.com

Alamat  
Jl. Mawar Putih No. 190, Waru Sidoarjo

Update

Place sticky footer content here.

Semisal kita coba edit datanya menjadi seperti berikut :

### Update Data Pegawai

Nama  
Anton Sucipto

Jenis Kelamin  
Pria

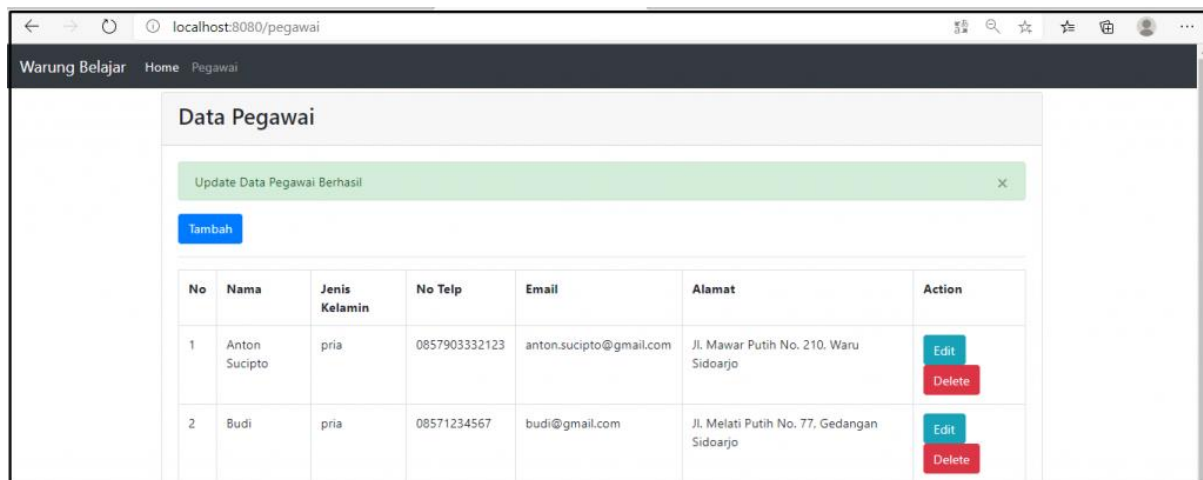
No Telp  
0857903332123

Email  
anton.sucipto@gmail.com

Alamat  
Jl. Mawar Putih No. 210, Waru Sidoarjo

Update

lalu kita klik tombol update...



maka ketika proses update pegawai berhasil, akan redirect ke halaman tampil data pegawai, dan data pegawai tersebut sudah terupdate sesuai dengan yang kita isikan di form edit, dan ada notifikasi bahwa **“Update Data Pegawai Berhasil”**.

---

## Membuat Crud – Delete Data Pegawai

Setelah anda belajar bagaimana cara proses membuat menu untuk menampilkan, menambahkan, dan mengupdate data pegawai.

dalam tutorial berikutnya kita akan belajar bagaimana membuat fitur delete data.

Kalau kita melihat di file view di [app/views/pegawai/index.php](#) pada **line 42** ada perintah untuk membuat tombol delete :

```
<a title="Delete" href="<?= base_url("pegawai/delete/$row->id_pegawai") ?>" class="btn btn-danger" onclick="return confirm('Apakah Anda yakin ingin menghapus data ?') ">Delete</a>
```

dalam perintah diatas kita buat tombol delete yang ketika di klik akan mengarah ke URL : [base\\_url/pegawai/delete/id-pegawai](#), sehingga sebelum kita membuat proses deletenya kita buat dulu routing untuk kebutuhan delete data ini, silahkan buka file [Routes.php](#) di folder [app/Config/Routes.php](#), lalu tambahkan perintah routing dibawah ini :

```
$routes->get('pegawai/delete/(:num)', 'Pegawai::delete/$1');
```

Keterangan :

Route diatas digunakan ketika ada yang mengakses URL

: `base_url/pegawai/delete/id-pegawai` maka akan diproses pada Controller `Pegawai` dengan method `delete`, dan method delete ini memiliki 1 parameter didalamnya.

dalam route ini juga saya menambahkan placeholder agar parameternya hanya diperbolehkan berupa numeric atau angka saja.

Setelah itu kita akan buat method `delete` di Controller `Pegawai`, sehingga code dari controller `Pegawai` menjadi sebagai berikut :

```
<?php
namespace App\Controllers;

use App\Models\PegawaiModel;

class Pegawai extends BaseController
{
    protected $pegawai;

    function __construct()
    {
        $this->pegawai = new PegawaiModel();
    }

    public function index()
    {
        $data['pegawai'] = $this->pegawai->findAll();
        return view('pegawai/index', $data);
    }

    public function create()
    {
        return view('pegawai/create');
    }

    public function store()
    {
        if (!$this->validate([
            'nama' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'jenis_kelamin' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
            'no_telp' => [
                'rules' => 'required',
                'errors' => [
                    'required' => '{field} Harus diisi'
                ]
            ],
        ]),
```

```

        'email' => [
            'rules' => 'required|valid_email',
            'errors' => [
                'required' => '{field} Harus diisi',
                'valid_email' => 'Email Harus Valid'
            ]
        ],
        'alamat' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
    ]), {
        session()->setFlashdata('error', $this->validator->listErrors());
        return redirect()->back()->withInput();
    }

    $this->pegawai->insert([
        'nama' => $this->request->getVar('nama'),
        'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
        'no_telp' => $this->request->getVar('no_telp'),
        'email' => $this->request->getVar('email'),
        'alamat' => $this->request->getVar('alamat')
    ]);
    session()->setFlashdata('message', 'Tambah Data Pegawai Berhasil');
    return redirect()->to('/pegawai');
}

function edit($id)
{
    $dataPegawai = $this->pegawai->find($id);
    if (empty($dataPegawai)) {
        throw new \CodeIgniter\Exceptions\PageNotFoundException('Data Pegawai Tidak ditemukan !');
    }
    $data['pegawai'] = $dataPegawai;
    return view('pegawai/edit', $data);
}

public function update($id)
{
    if (!$this->validate([
        'nama' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'jenis_kelamin' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'no_telp' => [
            'rules' => 'required',
            'errors' => [
                'required' => '{field} Harus diisi'
            ]
        ],
        'email' => [
            'rules' => 'required|valid_email',
            'errors' => [
                'required' => '{field} Harus diisi',
                'valid_email' => 'Email Harus Valid'
            ]
        ]
    ])) {

```

```

    ]
    ],
    'alamat' => [
        'rules' => 'required',
        'errors' => [
            'required' => '{field} Harus diisi'
        ]
    ],
    ],
    )) {
        session()->setFlashdata('error', $this->validator->listErrors());
        return redirect()->back();
    }

    $this->pegawai->update($id, [
        'nama' => $this->request->getVar('nama'),
        'jenis_kelamin' => $this->request->getVar('jenis_kelamin'),
        'no_telp' => $this->request->getVar('no_telp'),
        'email' => $this->request->getVar('email'),
        'alamat' => $this->request->getVar('alamat')
    ]);
    session()->setFlashdata('message', 'Update Data Pegawai Berhasil');
    return redirect()->to('/pegawai');
}

function delete($id)
{
    $dataPegawai = $this->pegawai->find($id);
    if (empty($dataPegawai)) {
        throw new \CodeIgniter\Exceptions\PageNotFoundException('Data Pegawai Tidak ditemukan !');
    }
    $this->pegawai->delete($id);
    session()->setFlashdata('message', 'Delete Data Pegawai Berhasil');
    return redirect()->to('/pegawai');
}
}

```

Keterangan :

Method delete ada pada line 139 – 147, pada method **delete** ini kita berikan **1 parameter** yaitu **variabel \$id**, dimana **variabel \$id** ini akan berisi **id-pegawai** yang akan di delete.

```
$dataPegawai = $this->pegawai->find($id);
```

berikutnya pada line 141 kita mencari **data pegawai** sesuai dengan **primary key**, sehingga kita menggunakan method **find** dengan menggunakan object **pegawai**.

```

if (empty($dataPegawai)) {
    throw new \CodeIgniter\Exceptions\PageNotFoundException('Data Pegawai Tidak ditemukan !');
}

```

berikutnya kita lakukan **pengecekan apakah ada data pegawai yang dicari tersebut**, jika tidak ada maka akan menjalankan baris code 143 dan menampilkan tulisan **"Data Pegawai Tidak ditemukan !"**

```
$this->pegawai->delete($id);
```



jika data pegawai ternyata ada maka akan menjalankan line 145, kita dapat menghapus data pegawai dengan menggunakan method **delete**, karena kita menghapus data pegawai menggunakan nilai pada kolom id\_pegawai yang merupakan kolom primary key.

```
session()->setFlashdata('message', 'Delete Data Pegawai Berhasil');  
return redirect()->to('/pegawai');
```

Line 146 digunakan untuk mengirimkan **flashdata** dengan menggunakan **session**, dengan value “Delete Data Pegawai Berhasil”, lalu berikutnya akan di redirect ke url **Pegawai.**, session flashdata ini digunakan untuk menampilkan **notifikasi** dibagian view **tampil data pegawai** saat proses delete data pegawai berhasil, pesan yang ditampilkan adalah “Delete Data Pegawai Berhasil”.

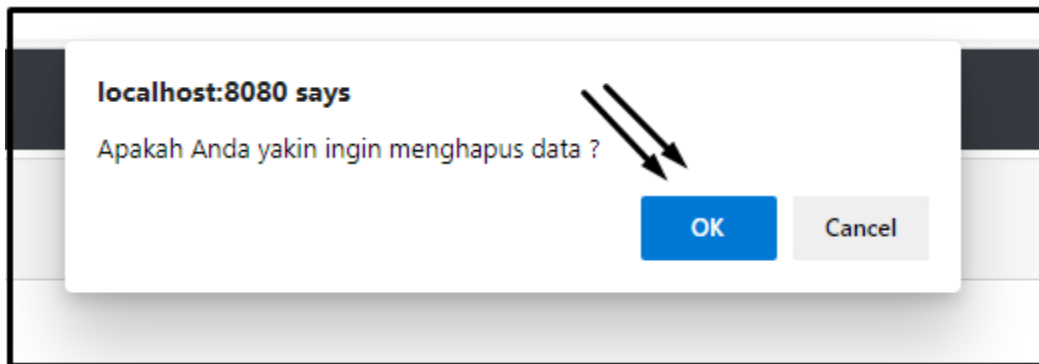
## Testing Proses Delete Data

Setelah kita buat fitur delete datanya, berikutnya kita akan coba delete data pegawai, kita akses terlebih dahulu data tampil pegawai, jangan lupa untuk jalankan local development server terlebih dahulu, jika sudah silahkan akses URL :

localhost:8080/pegawai

Data Pegawai						
<a href="#">Tambah</a>						
No	Nama	Jenis Kelamin	No Telp	Email	Alamat	Action
1	Anton Sucipto	pria	0857903332123	anton.sucipto@gmail.com	Jl. Mawar Putih No. 210, Waru Sidoarjo	<a href="#">Edit</a> <a href="#">Delete</a>
2	Budi	pria	08571234567	budi@gmail.com	Jl. Melati Putih No. 77, Gedangan Sidoarjo	<a href="#">Edit</a> <a href="#">Delete</a>
3	Dita	wanita	08122334455	dita@gmail.com	Jl. Rembulan No. 90, Krian Sidoarjo	<a href="#">Edit</a> <a href="#">Delete</a>

semisal kita akan menghapus data pegawai pertama dengan nama Anton Sucipto, klik tombol Delete..



akan tampil alert konfirmasi proses menghapus data, klik tombol OK.

Data Pegawai						
Delete Data Pegawai Berhasil						
Tambah						
No	Nama	Jenis Kelamin	No Telp	Email	Alamat	Action
1	Budi	pria	08571234567	budi@gmail.com	Jl. Melati Putih No. 77, Gedangan Sidoarjo	<div>Edit</div> <div>Delete</div>
2	Dita	wanita	08122334455	dita@gmail.com	Jl. Rembulan No. 90, Krian Sidoarjo	<div>Edit</div> <div>Delete</div>

jika proses delete data selesai.. maka data akan hilang, dan terdapat notifikasi bertuliskan "Delete Data Pegawai Berhasil".