

Panduan Praktikum Pemrograman Web – CodeIgniter4

Sumber Kode: github.com/AhmadMauludin/ci4

1. Unduh package codeigniter4, rename dengan nama_aplikasi dan simpan pada folder htdocs.
2. Buka file app/Config/Routes.php ubah kodennya dengan kode berikut (Routes.php)
3. Rename file env menjadi .env lalu ubah seluruh isinya dengan kode berikut (.env)
4. Pindahkan semua file yang ada dalam folder public ke root folder aplikasinya (dalam folder nama_aplikasi)
5. Buka file index.php yang sudah dipindah, ubah bagian `require FCPATH . './app/Config/Paths.php';` menjadi
`require FCPATH . 'app/Config/Paths.php';`
6. Buka file app/Config/App.php lalu ubah bagian '`http://localhost:8080/`'
Menjadi '`http://localhost:80/cruds2/`' atau '`http://localhost/cruds2/`' atau '`http://localhost:6969/cruds2/`'
7. Buka file app/Config/Paths.php tambahkan `public string $publicDirectory = __DIR__ . '/../';`
Di bawah `public string $viewDirectory = __DIR__ . '/../Views';`
8. Buat file baru (controller) bernama Barang.php pada folder app/Controllers isi dengan kode berikut (Barang.php)
9. Buat file baru (model) bernama Barang_model pada folder app/Models isikan dengan kode berikut (Barang_model.php)
10. Buat 2 file pada folder app/Views bernama header_view.php dan footer_view.php isikan dengan kode berikut (header_view.php dan footer_view.php)
11. Buat folder baru bernama barang di dalam folder app/Views
12. Buat 3 file baru pada folder barang tersebut berupa : barang_view.php edit_view.php dan tambah_view.php
Masukan kode berikut ke dalam file sesuai dengan nama filenya.
13. Buat database baru bernama ci4_barang.sql (atau sesuai nama database pada file .env)
14. Unduh file database bernama ci4_barang.sql lalu import ke dalam database yang sudah dibuat
15. Jalankan aplikasi di browser dengan memanggil localhost/nama_aplikasi

Penjelasan Kode Program

.env

File .env dalam CodeIgniter 4 digunakan untuk menyimpan konfigurasi lingkungan (environment) dan database secara aman. Berikut adalah penjelasan setiap bagian dari kode .env tersebut:

1. Menentukan Lingkungan Aplikasi

CI_ENVIRONMENT = development

- ◆ Fungsi:

- Menentukan mode aplikasi yang sedang berjalan.
- Nilai yang umum digunakan:
 - development → Untuk pengembangan, menampilkan error secara lengkap.
 - testing → Untuk pengujian, biasanya tanpa error output.
 - production → Untuk produksi, menyembunyikan error.

❖ Dalam mode "development":

- Semua error dan debug log akan ditampilkan.
 - Sangat berguna saat membuat dan memperbaiki aplikasi.
-

2. Konfigurasi Database

database.default.hostname = localhost

- ◆ Fungsi:

- Menentukan host database yang digunakan.
 - localhost berarti database berjalan di server lokal.
-

database.default.database = ci4_barang

- ◆ Fungsi:

- Menentukan nama database yang digunakan.
 - ci4_barang adalah nama database yang harus sudah dibuat di MySQL.
-

database.default.username = root

- ◆ Fungsi:

- Username untuk mengakses database.
 - root adalah user default di MySQL.
-

database.default.password =

◆ Fungsi:

- Password untuk user database.
 - Kosong berarti tanpa password (umumnya di MySQL lokal).
-

database.default.DBDriver = MySQLi

◆ Fungsi:

- Driver database yang digunakan.
 - MySQLi adalah driver untuk MySQL di PHP.
-

database.default.DBPrefix =

◆ Fungsi:

- Prefix (awalan) untuk tabel dalam database.
- Kosong berarti tidak menggunakan prefix.
- Bisa diisi misalnya:
- database.default.DBPrefix = ci_

Sehingga tabel users menjadi ci_users.

database.default.port = 3306

◆ Fungsi:

- Port yang digunakan oleh MySQL.
 - 3306 adalah port default MySQL.
-

Routes.php

Kode yang diberikan adalah bagian dari konfigurasi routing pada framework CodeIgniter 4. Routing ini menentukan bagaimana URL yang diakses akan diarahkan ke controller dan metode tertentu. Berikut penjelasan untuk setiap bagiannya:

1. Use Statement

```
use CodeIgniter\Router\RouteCollection;
```

- Ini adalah deklarasi penggunaan (use) untuk class RouteCollection, yang berfungsi untuk mengelola rute dalam aplikasi CodeIgniter.
-

2. Variabel \$routes

```
/**
```

```
* @var RouteCollection $routes
```

*/

- Komentar ini (@var RouteCollection \$routes) digunakan sebagai anotasi untuk membantu editor kode memahami bahwa \$routes adalah instance dari RouteCollection.
-

3. Rute "Halo"

```
$routes->get('halo', 'Halo::index');
```

- Mendefinisikan rute GET dengan path halo, yang akan diarahkan ke controller Halo dan method index.
 - Artinya, jika pengguna mengakses <http://example.com/halo>, maka request akan diproses oleh method index() di dalam Halo controller.
-

4. Rute Home (Barang)

```
$routes->get('/', 'Barang::index');
```

```
$routes->get('barang', 'Barang::index');
```

- Rute pertama ('/') adalah **rute utama** (home), yang akan memanggil method index() dari controller Barang.
 - Rute kedua ('barang') juga memanggil method index() dari controller Barang.
 - Jadi, jika pengguna mengakses <http://localhost:8080/> atau <http://localhost:8080/barang>, maka method index() dari Barang controller akan dijalankan.
-

5. Rute Halaman Tambah Barang

```
$routes->get('barang/tambah', 'Barang::tambah');
```

- Jika pengguna mengakses <http://localhost:8080/barang/tambah>, maka method tambah() dalam Barang controller akan dijalankan.
 - ini digunakan untuk menampilkan formulir tambah barang.
-

6. Rute Halaman Edit Barang

```
$routes->get('barang/edit/(:any)', 'Barang::edit/$1');
```

- Rute ini mengarah ke method edit() dalam Barang controller.
 - (:any) adalah wildcard yang menangkap parameter apapun dari URL, dan diteruskan sebagai parameter ke method edit().
 - \$1 adalah placeholder untuk parameter pertama yang diterima.
 - Misalnya:
 - <http://localhost:8080/barang/edit/5> → edit(5) dipanggil.
-

7. Rute Proses CRUD

Insert (Tambah Barang)

```
$routes->post('barang/add', 'Barang::add');
```

- Menangani request POST untuk http://localhost:8080/barang/add.
- Memanggil method add() dalam controller Barang.
- Digunakan untuk menyimpan data barang baru ke database.

Update (Perbarui Barang)

```
$routes->post('barang/update', 'Barang::update');
```

- Menangani request POST untuk http://localhost:8080/barang/update.
- Memanggil method update() dalam controller Barang.
- Digunakan untuk memperbarui data barang di database.

Hapus Barang

```
$routes->get('barang/hapus/(:any)', 'Barang::hapus/$1');
```

- Menangani request GET untuk http://localhost:8080/barang/hapus/{id}.
- (:any) menangkap ID barang yang akan dihapus.
- Misalnya:
 - http://localhost:8080/barang/hapus/10 → method hapus(10) dipanggil di controller Barang.

Barang.php (controller)

Kode ini adalah controller dalam framework CodeIgniter 4 yang bertanggung jawab untuk mengelola data barang, termasuk menampilkan daftar barang, menambahkan, mengedit, memperbarui, dan menghapus barang.

Penjelasan Setiap Bagian

1. Namespace dan Import Kelas

```
namespace App\Controllers;  
use CodeIgniter\Controller;  
use App\Models\Barang_model;  
  
• namespace App\Controllers;  
  → Menandakan bahwa file ini berada di dalam namespace App\Controllers.  
• use CodeIgniter\Controller;  
  → Mengimpor Controller dari CodeIgniter sebagai kelas induk.  
• use App\Models\Barang_model;  
  → Mengimpor model Barang_model yang akan digunakan untuk mengakses database.
```

2. Mendefinisikan Kelas Barang

```
class Barang extends Controller
```

- Kelas Barang adalah controller yang menangani operasi terkait barang.
- Mewarisi Controller dari CodeIgniter agar bisa menggunakan fitur bawaan, seperti request dan response.

3. Method index() – Menampilkan Data Barang

```
public function index()  
{  
    $model = new Barang_model;  
    $data['title'] = 'Data Barang';  
    $data['getBarang'] = $model->getBarang();  
    echo view('header_view', $data);  
    echo view('barang/barang_view', $data);  
    echo view('footer_view', $data);  
}
```

- Membuat objek Barang_model untuk mengambil data dari database.
- \$data['title'] → Menyimpan judul halaman.
- \$data['getBarang'] → Menyimpan daftar barang dari database (getBarang()).
- Menampilkan tampilan (view)
 - header_view → Header halaman.
 - barang_view → Isi utama halaman yang menampilkan daftar barang.
 - footer_view → Footer halaman.

4. Method tambah() – Menampilkan Form Tambah Barang

```
public function tambah()  
{  
    $data['title'] = 'Tambah Data Barang';  
    echo view('header_view', $data);  
    echo view('barang/tambah_view', $data);  
    echo view('footer_view', $data);  
}
```

- Menampilkan halaman untuk menambahkan barang.
- Menggunakan tampilan:
 - header_view
 - tambah_view (form input barang baru)
 - footer_view

5. Method add() – Menyimpan Data Barang Baru

```

public function add()
{
    $model = new Barang_model;
    $data = array(
        'nama_barang' => $this->request->getPost('nama'),
        'qty'      => $this->request->getPost('qty'),
        'harga_beli' => $this->request->getPost('beli'),
        'harga_jual' => $this->request->getPost('jual')
    );
    $model->saveBarang($data);
    echo '<script>
        alert("Sukses Tambah Data Barang");
        window.location=' . base_url('barang') .
    </script>';
}


- Membuat objek Barang_model.
- Mengambil data dari form ($_POST) menggunakan getPost().
- Menyimpan data dengan memanggil saveBarang($data).
- Menampilkan alert JavaScript untuk konfirmasi sukses dan mengarahkan pengguna ke halaman barang.

```

6. Method edit(\$id) – Menampilkan Form Edit Barang

```

public function edit($id)
{
    $model = new Barang_model;
    $getBarang = $model->getBarang($id)->getRow();
    if (isset($getBarang)) {
        $data['barang'] = $getBarang;
        $data['title'] = 'Edit ' . $getBarang->nama_barang;

        echo view('header_view', $data);
        echo view('barang/edit_view', $data);
        echo view('footer_view', $data);
    } else {
        echo '<script>

```

```

        alert("ID barang ' . $id . ' Tidak ditemukan");

        window.location="" . base_url('barang') . ""

    </script>';

}

}

```

- Mengambil data barang berdasarkan ID menggunakan getBarang(\$id).
 - Jika barang ditemukan, maka:
 - Menyimpan data dalam \$data['barang'].
 - Menampilkan form edit (edit_view).
 - Jika barang tidak ditemukan, menampilkan alert JavaScript.
-

7. Method update() – Menyimpan Perubahan Data Barang

```

public function update()

{
    $model = new Barang_model;

    $id = $this->request->getPost('id_barang');

    $data = array(
        'nama_barang' => $this->request->getPost('nama'),
        'qty'      => $this->request->getPost('qty'),
        'harga_beli' => $this->request->getPost('beli'),
        'harga_jual' => $this->request->getPost('jual')
    );

    $model->editBarang($data, $id);

    echo '<script>
        alert("Sukses Edit Data Barang");
        window.location="" . base_url('barang') . ""
    </script>';

}

```

- Mengambil ID barang dari form (\$_POST).
 - Mengambil data baru dari form (nama, qty, harga_beli, harga_jual).
 - Memanggil method editBarang(\$data, \$id) untuk menyimpan perubahan ke database.
 - Menampilkan alert JavaScript.
-

8. Method hapus(\$id) – Menghapus Barang

```

public function hapus($id)
{
    $model = new Barang_model;
    $getBarang = $model->getBarang($id)->getRow();
    if (isset($getBarang)) {
        $model->hapusBarang($id);
        echo '<script>
            alert("Hapus Data Barang Sukses");
            window.location="'. base_url('barang') .'"
        </script>';
    } else {
        echo '<script>
            alert("Hapus Gagal !, ID barang ' . $id . ' Tidak ditemukan");
            window.location="'. base_url('barang') .'"
        </script>';
    }
}

```

- Mengambil data barang berdasarkan ID.
 - Jika barang ditemukan:
 - Menghapus data dengan hapusBarang(\$id).
 - Menampilkan alert JavaScript sukses.
 - Jika barang tidak ditemukan, menampilkan alert error.
-

Barang_Model.php (model)

Kode di bawah ini merupakan model dalam framework CodeIgniter 4 yang digunakan untuk mengelola data barang dalam database. Model ini mengatur operasi CRUD (Create, Read, Update, Delete) pada tabel barang.

Penjelasan Setiap Bagian Kode

1. Namespace dan Penggunaan Model

```
namespace App\Models;
```

```
use CodeIgniter\Model;
```

- namespace App\Models; → Menentukan bahwa class ini berada dalam namespace App\Models.
 - use CodeIgniter\Model; → Mengimpor Model dari CodeIgniter untuk digunakan sebagai dasar (extends) dalam class ini.
-

2. Deklarasi Class

```
class Barang_model extends Model
```

- Barang_model adalah nama class yang berfungsi sebagai model untuk tabel barang.
 - extends Model berarti class ini mewarisi semua fitur dari Model CodeIgniter.
-

3. Properti \$table

```
protected $table = 'barang';
```

- Menentukan bahwa model ini akan berinteraksi dengan tabel barang dalam database.
-

4. Method getBarang(\$id = false)

```
public function getBarang($id = false)
```

```
{
```

```
    if ($id === false) {  
        return $this->findAll();  
    } else {  
        return $this->getWhere(['id_barang' => $id]);  
    }  
}
```

- **Fungsi:** Mengambil data dari tabel barang.
- **Parameter:** \$id (opsional) → jika diberikan, akan mengambil data berdasarkan ID barang.
- **Penjelasan:**
 - Jika tidak ada ID yang diberikan (\$id === false), maka akan mengambil semua data barang dengan findAll().
 - Jika ID diberikan, maka hanya mengambil data barang dengan ID tertentu menggunakan getWhere(['id_barang' => \$id]).

5. Method saveBarang(\$data)

```
public function saveBarang($data)
```

```
{
```

```
    $builder = $this->db->table($this->table);  
    return $builder->insert($data);  
}
```

- **Fungsi:** Menyimpan data barang baru ke database.
- **Parameter:** \$data → array berisi data barang yang akan disimpan.
- **Penjelasan:**
 - \$builder = \$this->db->table(\$this->table); → Mengakses tabel barang.

- \$builder->insert(\$data); → Menyisipkan (INSERT) data ke tabel.

6. Method editBarang(\$data, \$id)

```
public function editBarang($data, $id)
```

```
{
```

```
    $builder = $this->db->table($this->table);
    $builder->where('id_barang', $id);
    return $builder->update($data);
```

```
}
```

- **Fungsi:** Mengedit atau memperbarui data barang berdasarkan ID.
- **Parameter:**
 - \$data → Array berisi data yang akan diperbarui.
 - \$id → ID barang yang akan diperbarui.
- **Penjelasan:**
 - \$builder = \$this->db->table(\$this->table); → Mengakses tabel barang.
 - \$builder->where('id_barang', \$id); → Menentukan barang mana yang akan diupdate berdasarkan id_barang.
 - \$builder->update(\$data); → Memperbarui data yang ada di database.

7. Method hapusBarang(\$id)

```
public function hapusBarang($id)
```

```
{
```

```
    $builder = $this->db->table($this->table);
    return $builder->delete(['id_barang' => $id]);
```

```
}
```

- **Fungsi:** Menghapus data barang dari database berdasarkan ID.
- **Parameter:** \$id → ID barang yang akan dihapus.
- **Penjelasan:**
 - \$builder = \$this->db->table(\$this->table); → Mengakses tabel barang.
 - \$builder->delete(['id_barang' => \$id]); → Menghapus barang dengan id_barang yang sesuai.

barang_view.php

Kode di bawah merupakan bagian dari tampilan halaman yang menampilkan **daftar barang** dalam sebuah tabel menggunakan **Bootstrap** untuk styling dan PHP untuk menampilkan data dari database. Berikut adalah penjelasan setiap bagian dari kode ini:

1. Struktur Kontainer

```
<div class="container pt-5">
```

- <div class="container"> → Bootstrap class untuk membuat kontainer responsif.
 - pt-5 → Bootstrap class yang memberikan padding-top sebesar 5 (spasi di atas elemen).
-

2. Tombol "Tambah Data"

```
<a href="= base_url('barang/tambah'); ?" class="btn btn-success mb-2">Tambah Data</a>
```

- → Link menuju halaman Tambah Data.
 - base_url('barang/tambah') → Fungsi CodeIgniter untuk mendapatkan URL halaman tambah barang.
 - class="btn btn-success mb-2":
 - btn btn-success → Styling tombol hijau (Bootstrap).
 - mb-2 → Margin bawah sebesar 2 untuk memberi spasi.
-

3. Kartu (Card) untuk Menampilkan Data

```
<div class="card">
```

- <div class="card"> → Bootstrap class untuk membuat kotak (card) yang membungkus tabel.
-

4. Header Card

```
<div class="card-header bg-info text-white">  
  <h4 class="card-title">Data Barang</h4>  
</div>
```

- <div class="card-header bg-info text-white"> → Header card dengan background biru (bg-info) dan teks putih (text-white).
 - <h4 class="card-title">Data Barang</h4> → Judul dalam header card.
-

5. Body Card

```
<div class="card-body">
```

- <div class="card-body"> → Area isi dari card yang akan berisi tabel data barang.
-

6. Tabel Data Barang

```
<div class="table-responsive">
```

```
  <table class="table table-bordered table-striped">
```

- <div class="table-responsive"> → Membuat tabel bisa di-scroll saat layar kecil.

- <table class="table table-bordered table-striped">:
 - table → Bootstrap class untuk tabel.
 - table-bordered → Tabel dengan border.
 - table-striped → Tabel dengan efek warna bergantian di setiap baris.
-

7. Header Tabel

```
<thead>  
  <tr>  
    <th>No.</th>  
    <th>Nama Barang</th>  
    <th>Qty</th>  
    <th>Harga Beli</th>  
    <th>Harga Jual</th>  
    <th>Aksi</th>  
  </tr>  
</thead>
```

- Menentukan kolom dalam tabel:

- No. → Nomor urut barang.
- Nama Barang → Nama barang.
- Qty → Jumlah barang.
- Harga Beli → Harga beli barang.
- Harga Jual → Harga jual barang.
- Aksi → Tombol Edit dan Hapus.

8. Menampilkan Data Barang (Looping)

```
<?php $no = 1;  
foreach ($getBarang as $isi) { ?>  
  • $getBarang → Variabel yang berisi daftar barang dari database.  
  • foreach ($getBarang as $isi) → Melakukan loop pada setiap barang dalam $getBarang.  
  • $no = 1; → Inisialisasi nomor urut.
```

9. Menampilkan Data Setiap Baris

```
<tr>  
  <td><?= $no; ?></td>
```

```

<td><?= $isi['nama_barang']; ?></td>
<td><?= $isi['qty']; ?></td>
<td>Rp. <?= number_format($isi['harga_beli']); ?>,-</td>
<td>Rp. <?= number_format($isi['harga_jual']); ?>,-</td>
<td>


- <td><?= $no; ?></td> → Menampilkan nomor urut.
- <td><?= $isi['nama_barang']; ?></td> → Menampilkan nama barang.
- <td><?= $isi['qty']; ?></td> → Menampilkan jumlah barang.
- <td>Rp. <?= number_format($isi['harga_beli']); ?>,-</td> → Menampilkan harga beli, diformat dengan number_format().
- <td>Rp. <?= number_format($isi['harga_jual']); ?>,-</td> → Menampilkan harga jual, juga diformat.

```

10. Tombol Edit dan Hapus

```

<a href="= base_url('barang/edit/' . $isi['id_barang']); ?&gt;" class="btn btn-success"&gt;Edit&lt;/a&gt;
&lt;a href="<?= base_url('barang/hapus/' . $isi['id_barang']); ?&gt;" onclick="javascript:return confirm('Apakah ingin menghapus data barang ?')" class="btn btn-danger"&gt;Hapus&lt;/a&gt;
</pre

```

- Tombol Edit
 - " class="btn btn-success">Edit
<ul - **base_url('barang/edit/' . \$isi['id_barang'])** → URL menuju halaman edit barang berdasarkan id_barang.
 - **btn btn-success** → Tombol hijau (Bootstrap).
- Tombol Hapus
 - " onclick="javascript:return confirm('Apakah ingin menghapus data barang ?')" class="btn btn-danger">Hapus
<ul - **base_url('barang/hapus/' . \$isi['id_barang'])** → URL untuk menghapus barang berdasarkan id_barang.
 - **onclick="javascript:return confirm('Apakah ingin menghapus data barang ?')"** → Menampilkan konfirmasi sebelum menghapus barang.
 - **btn btn-danger** → Tombol merah (Bootstrap).
-

11. Akhir dari Looping

```

<?php $no++; ?>


- $no++ → Menambah nomor urut untuk barang berikutnya.
- } → Akhir dari perulangan foreach.

```

edit_view.php

Berikut adalah penjelasan setiap bagian dalam kode HTML + PHP yang digunakan untuk halaman edit data barang dalam aplikasi berbasis CodeIgniter:

1. Container (Bagian Utama Halaman)

```
<div class="container p-5">
```

- div ini adalah container utama untuk membungkus seluruh elemen dalam halaman.
 - Menggunakan Bootstrap class:
 - container → Membuat layout responsif dengan margin otomatis.
 - p-5 → Memberikan padding (jarak dalam) sebesar 5 unit.
-

2. Tombol Kembali

```
<a href="= base_url('barang'); ?" class="btn btn-secondary mb-2">Kembali</a>
```

- Tombol kembali yang mengarah ke halaman utama daftar barang (base_url('barang')).
 - Menggunakan class Bootstrap:
 - btn btn-secondary → Tombol dengan warna abu-abu.
 - mb-2 → Memberikan margin bawah sebesar 2 unit agar tidak terlalu dekat dengan elemen di bawahnya.
-

3. Kartu (Card) untuk Formulir Edit

```
<div class="card">
```

- Membungkus seluruh form dalam card Bootstrap agar tampil lebih rapi.

3.1. Header Card

```
<div class="card-header bg-info text-white">  
    <h4 class="card-title">Edit Barang : <?= $barang->nama_barang; ?></h4>  

```

- Bagian header untuk card yang menampilkan judul "Edit Barang".
 - <?= \$barang->nama_barang; ?> → Menampilkan nama barang yang sedang diedit.
 - bg-info → Latar belakang header berwarna biru.
 - text-white → Teks pada header berwarna putih.
-

4. Formulir Edit Data Barang

```
<form method="post" action="= base_url('barang/update'); ?&gt;"&gt;</pre
```

- Formulir untuk mengedit data barang.
- method="post" → Menggunakan metode POST untuk mengirim data ke server.

- action="= base_url('barang/update'); ?" → Data dikirim ke URL barang/update, yang akan menangani pembaruan data barang di server.
-

5. Input untuk Nama Barang

```
<div class="form-group">  
    <label for="">Nama Barang</label>  
    <input type="text" value="= $barang-&gt;nama_barang; ?" name="nama" required class="form-control">  
</div>
```

- Input teks untuk mengedit nama barang.
 - value="= \$barang->nama_barang; ?" → Mengisi input dengan nama barang yang sudah ada sebelumnya.
 - name="nama" → Nama field yang akan dikirim ke server.
 - required → Wajib diisi sebelum dikirim.
 - class="form-control" → Menggunakan Bootstrap agar tampilan input lebih rapi.
-

6. Input untuk Quantity (Qty)

```
<div class="form-group">  
    <label for="">Qty</label>  
    <input type="number" value="= $barang-&gt;qty; ?" name="qty" required class="form-control">  
</div>
```

- Input angka untuk mengedit jumlah barang (Qty).
 - type="number" → Hanya bisa diisi dengan angka.
 - value="= \$barang->qty; ?" → Mengisi input dengan nilai qty sebelumnya.
-

7. Input untuk Harga Beli

```
<div class="form-group">  
    <label for="">Harga Beli</label>  
    <input type="number" value="= $barang-&gt;harga_beli; ?" name="beli" required class="form-control">  
</div>
```

- Input angka untuk mengedit harga beli barang.

8. Input untuk Harga Jual

```
<div class="form-group">  
    <label for="">Harga Jual</label>  
    <input type="number" value="= $barang-&gt;harga_jual; ?" name="jual" required class="form-control">
```

```
</div>
```

- Input angka untuk mengedit harga jual barang.
-

9. Input Tersembunyi untuk ID Barang

```
<input type="hidden" value=<?= $barang->id_barang; ?>" name="id_barang">
```

- Input tersembunyi (hidden input) untuk mengirimkan id_barang tanpa terlihat oleh pengguna.
 - name="id_barang" → Supaya ID barang tetap dikirim ke server untuk referensi pembaruan.
-

10. Tombol Simpan Perubahan

```
<button class="btn btn-success">Edit Data</button>
```

- Tombol untuk menyimpan perubahan yang dilakukan pada formulir.
 - btn btn-success → Menggunakan warna hijau untuk tombol.
-

edit_view.php

File ini digunakan untuk menampilkan formulir tambah data barang dalam aplikasi berbasis CodeIgniter dan Bootstrap. Berikut penjelasan setiap bagiannya:

1. Container (Bagian Utama Halaman)

```
<div class="container p-5">
```

- div ini berfungsi sebagai pembungkus utama halaman.
 - Menggunakan Bootstrap:
 - container → Membuat tata letak responsif dengan margin otomatis.
 - p-5 → Memberikan padding (jarak dalam) sebesar 5 unit agar elemen tidak terlalu rapat.
-

2. Tombol Kembali

```
<a href=<?= base_url('barang'); ?>" class="btn btn-secondary mb-2">Kembali</a>
```

- Tombol untuk kembali ke halaman daftar barang.
 - base_url('barang') → Mengarahkan pengguna kembali ke halaman utama barang.
 - Menggunakan Bootstrap:
 - btn btn-secondary → Tombol dengan warna abu-abu.
 - mb-2 → Memberikan margin bawah 2 unit agar tidak terlalu dekat dengan elemen berikutnya.
-

3. Kartu (Card) untuk Formulir

- ```
<div class="card">
 • Membungkus formulir dalam kartu Bootstrap agar tampilan lebih rapi.
```

### 3.1. Header Card

```
<div class="card-header bg-info text-white">
 <h4 class="card-title">Tambah Data Barang</h4>
```

- ```
</div>
  • Bagian header dari kartu yang berisi judul halaman: "Tambah Data Barang".
  • Bootstrap class:
    ○ bg-info → Memberikan latar belakang warna biru.
    ○ text-white → Membuat teks di header menjadi putih.
```
-

4. Formulir Tambah Data Barang

```
<form method="post" action="<?= base_url('barang/add'); ?>">
```

- Formulir untuk menambahkan data barang baru.
 - method="post" → Data akan dikirim ke server menggunakan metode POST.
 - action="<?= base_url('barang/add'); ?>" → Data dikirim ke barang/add, yang akan menangani proses penyimpanan data di server.
-

5. Input untuk Nama Barang

```
<div class="form-group">
  <label for="">Nama Barang</label>
  <input type="text" name="nama" class="form-control" required>
</div>
```

- Input teks untuk mengisi nama barang.
 - name="nama" → Nama field yang dikirim ke server.
 - class="form-control" → Menggunakan Bootstrap agar tampilan lebih rapi.
 - required → Wajib diisi sebelum formulir bisa dikirim.
-

6. Input untuk Quantity (Qty)

```
<div class="form-group">
  <label for="">Qty</label>
  <input type="number" name="qty" class="form-control" required>
</div>
```

- Input angka untuk jumlah barang yang akan ditambahkan.

- type="number" → Hanya bisa diisi dengan angka.
 - name="qty" → Nama field yang dikirim ke server.
-

7. Input untuk Harga Beli

```
<div class="form-group">  
    <label for="">Harga Beli</label>  
    <input type="number" name="beli" class="form-control" required>  
</div>
```

- Input angka untuk harga beli barang.
-

8. Input untuk Harga Jual

```
<div class="form-group">  
    <label for="">Harga Jual</label>  
    <input type="number" name="jual" class="form-control" required>  
</div>
```

- Input angka untuk harga jual barang.
-

9. Tombol Simpan Data

```
<button class="btn btn-success">Tambah Data</button>
```

- Tombol untuk mengirimkan formulir.
- btn btn-success → Menggunakan warna hijau untuk tombol.

Header_view.php

Kode ini adalah bagian untuk menampilkan halaman header / navbar. Berikut adalah penjelasan setiap bagiannya:

1. Deklarasi Doctype

```
<!doctype html>  
<html lang="en">
```

- <!doctype html> → Menentukan bahwa dokumen ini adalah HTML5.
- <html lang="en"> → Menentukan bahasa utama halaman adalah Bahasa Inggris.

2. Bagian <head> (Metadata dan Styling)

2.1. Judul Halaman

```
<title><?= $title;?></title>
```

- Menampilkan judul halaman yang berasal dari variabel \$title di CodeIgniter.
 - <?= \$title;?> → PHP short tag untuk mencetak isi variabel \$title.
-

2.2. Meta Tags (Pengaturan Halaman)

```
<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

- <meta charset="utf-8"> → Menggunakan UTF-8 agar bisa menampilkan karakter dari berbagai bahasa.
- <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
 - Membuat tampilan responsif.
 - width=device-width → Mengatur lebar sesuai layar perangkat.
 - initial-scale=1 → Menentukan zoom awal.
 - shrink-to-fit=no → Mencegah browser mengubah ukuran tampilan secara otomatis.

2.3. Import Bootstrap 4.3.1

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"  
crossorigin="anonymous">
```

- Menggunakan Bootstrap 4.3.1 dari CDN (Content Delivery Network).
 - integrity → Memastikan file yang diunduh tidak berubah dari sumber aslinya.
 - crossorigin="anonymous" → Mengizinkan pengambilan file dari domain lain dengan aman.
-

3. Bagian <body> (Konten Halaman)

```
<body>
```

Menandakan bahwa konten halaman dimulai dari sini.

4. Navigasi (Navbar)

```
<nav class="navbar navbar-expand-sm navbar-dark bg-info">
```

- Menggunakan Bootstrap untuk navbar.
 - navbar → Elemen navbar utama.
 - navbar-expand-sm → Navbar akan diperluas di layar berukuran $\geq 576\text{px}$ (small - sm).
 - navbar-dark → Teks navbar berwarna putih.
 - bg-info → Navbar berwarna biru muda.
-

4.1. Container dalam Navbar

```
<div class="container">
```

- Membungkus elemen navbar dalam container Bootstrap agar lebih rapi.
-

4.2. Logo/Brand Navbar

```
<a class="navbar-brand" href="<?= base_url();?>">Data Barang Toko Codekop</a>
```

- navbar-brand → Menampilkan nama Toko Codekop sebagai logo/brand navbar.
 - href="<?= base_url(); ?>" → Mengarahkan ke halaman utama aplikasi.
-

4.3. Tombol Hamburger Menu (Untuk Mobile)

```
<button class="navbar-toggler d-lg-none" type="button" data-toggle="collapse"
data-target="#collapsibleNavId" aria-controls="collapsibleNavId"
aria-expanded="false" aria-label="Toggle navigation"></button>
```

- Tombol menu (hamburger menu) yang muncul di layar kecil.
 - navbar-toggler → Membuat tombol menu dropdown untuk tampilan mobile.
 - d-lg-none → Tombol hanya muncul jika ukuran layar lebih kecil dari lg ($\geq 992\text{px}$).
 - data-toggle="collapse" data-target="#collapsibleNavId" → Ketika ditekan, akan membuka/tutup menu dengan id="collapsibleNavId".
 - aria-controls → Memberitahu screen reader bahwa tombol ini mengontrol collapsibleNavId.
 - aria-expanded="false" → Menunjukkan bahwa menu awalnya tersembunyi.
 - aria-label="Toggle navigation" → Label untuk aksesibilitas.
-

footer_view.php

Kode ini berisi JavaScript opsional yang diperlukan untuk mendukung fitur interaktif Bootstrap. Berikut adalah penjelasan setiap bagianya:

1. Komentar (Penjelasan Kode)

```
<!-- Optional JavaScript -->
```

```
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
```

- Memberikan informasi bahwa JavaScript ini opsional, tetapi sangat disarankan untuk fitur interaktif Bootstrap.
- Urutan file JavaScript:
 1. jQuery (wajib untuk Bootstrap).
 2. Popper.js (untuk tooltip dan dropdown).

3. Bootstrap JavaScript.

2. Memuat jQuery (Slim Version)

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"  
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"  
crossorigin="anonymous"></script>
```

Fungsi:

- Memuat jQuery versi 3.3.1 (Slim Version) dari CDN jQuery.
- Versi "Slim" berarti:
 - Tidak memiliki fitur AJAX dan efek animasi.
 - Hanya untuk manipulasi DOM dan event handling.

Atribut tambahan:

- integrity → Memastikan file yang diunduh tidak diubah dari sumber asli.
 - crossorigin="anonymous" → Mengizinkan pemuatannya dari domain lain dengan aman.
-

3. Memuat Popper.js

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"  
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"  
crossorigin="anonymous"></script>
```

Fungsi:

- Memuat Popper.js versi 1.14.7 dari CDN Cloudflare.
- Popper.js digunakan oleh Bootstrap untuk tooltip dan dropdown menu.

Atribut tambahan:

- integrity dan crossorigin="anonymous" sama seperti sebelumnya.
-

4. Memuat Bootstrap JavaScript

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"  
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYolly6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"  
crossorigin="anonymous"></script>
```

Fungsi:

- Memuat Bootstrap 4.3.1 JavaScript dari CDN StackPath.
- Mengaktifkan fitur Bootstrap seperti:
 - Modal
 - Dropdown

- Carousel
- Tooltip dan Popover (bergantung pada Popper.js)

Atribut tambahan:

- integrity dan crossorigin="anonymous" tetap sama.
-

5. Penutupan Body dan HTML

```
</body>
```

```
</html>
```

- </body> → Menutup elemen body, mengakhiri konten halaman.
 - </html> → Menutup elemen html, mengakhiri dokumen HTML.
-