DOKUMENTASI

Dokumen ini berisi tentang proses pembuatan project tes programmer dari awal hingga akhir. Dokumentasi ini diharapkan memberikan panduan yang jelas tentang langkah langkah proses pembuatannya. Jika terdapat kesalahan atau kekurangan dalam penulisan dokumen ini, penulis mengucapkan permohonan maaf dan terbuka untuk saran serta masukan guna perbaikan di masa mendatang.

1. Pembuatan Database

Database yang diperlukan terdiri atas 3 tabel, yaitu : produk, kategori, dan status. Tabel-tabel tersebut memiliki struktur:

- Tabel produk berisi id_produk, nama_produk, harga, id_kategori, dan id_status.
- Tabel kategori berisi id_kategori, dan nama_kategori
- Tabel status berisi id_status, dan nama_status

Dengan informasi tersebut, maka database dapat dibuat dengan perintah di bawah ini.

```
CREATE DATABASE test_program;
USE test_program;
CREATE TABLE kategori (
   id_kategori INT AUTO_INCREMENT PRIMARY KEY,
   nama_kategori VARCHAR(255) NOT NULL
);
CREATE TABLE status (
   id_status INT AUTO_INCREMENT PRIMARY KEY,
   nama_status VARCHAR(50) NOT NULL
);
CREATE TABLE produk (
   id_produk INT AUTO_INCREMENT PRIMARY KEY,
   nama_produk VARCHAR(255) NOT NULL,
   harga INT NOT NULL,
   id_kategori INT,
   id_status INT,
   FOREIGN KEY (id_kategori) REFERENCES kategori(id_kategori),
   FOREIGN KEY (id_status) REFERENCES status(id_status)
);
INSERT INTO status (nama_status) VALUES ('bisa dijual'), ('tidak bisa dijual');
```

Foreign key dihubungkan pada id_kategori dan id_status sebagai penghubung antar table.

Setelah database dibuat, maka selanjutnya mengatur config database pada file project test_programmer. File tersebut terletak pada application/config/database.php. yang perlu diperhatikan adalah konfigurasi username dan password disesuaikan dengan settingan database, serta nama database yang digunakan. Saya juga mengghilangkan pengaturan 'failover' => array() untuk menghindari notifikasi error yang muncul saat awal membuka halaman website.

```php
$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'test_program',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'save_queries' => TRUE
);
```

Gambar 1.1 konfigurasi file database.php

2. Pengambilan Data dari API

Sebelum memulai pembuatan website, saya ditugaskan untuk mengambil data yang disediakan pada link API : **https://recruitment.fastprint.co.id/tes/api_tes_programmer**. Untuk mengambil data tersebut, diperlukan autentikasi yang cukup ketat dengan username dan password yang berubah ubah menyesuaikan waktu server. Terdapat contoh username dan password pada link tes programmer yang diberikan. Untuk username menggunakan format:

tesprogrammer(2 digit tanggal sekarang)(2 digit bulan sekarang)(2 digit tahun sekarang)C(2 digit jam sekarang)

Setelah saya amati, penulis mengambil kesimpulan bahwa server menggunakan format waktu UTC +8 karena terdapat perbedaan jam pada device yang saya gunakan (UTC+7). Lalu untuk password nya menggunakan algoritma MD5(Message-Direct Algoritm 5) sebagai keamanannya. Untuk password menggunakan format:

MD5 dari : bisacoding-(2 digit tanggal sekarang)-(2 digit bulan sekarang)-(2 digit tahun sekarang)

Dengan informasi username dan password telah didapatkan, maka dapat mengkonfigurasi perintah fungsi ambil_data_API(). Fungsi ini dibuat di dalam file application/controllers/Produk.php. zona waktu diatur Asia/Makassar agar mengikuti waktu UTC+8 seperti server API.

```php
public function ambil_data_api(){
    date_default_timezone_set('Asia/Makassar');

    $url ="https://recruitment.fastprint.co.id/tes/api_tes_programmer";

    $tanggal = date('d');
    $bulan = date('m');
    $tahun = date('y');
    $jam = date ('H');

    $username = "tesprogrammer{$tanggal}{$bulan}{$tahun}C{$jam}";
    $password = md5("bisacoding-{$tanggal}-{$bulan}-{$tahun}");
```

Gambar 2.1 pengaturan username dan password pada Produk.php

Selanjutnya mengatur cURL untuk mengirim username dan password. Username dan password dikirim dalam array agar tidak terjadi error saat pengiriman data.

```php
$data = array(
    'username' => $username,
    'password' => $password
);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$response = curl_exec($ch);
curl_close($ch);
```

Gambar 2.2 pengaturan cURL pada Produk.php

Data yang didapatkan kemudian dimasukkan ke dalam database dengan perintah pada gambar 2.3. konfigurasi juga diberikan output respon untuk mengetahui apakah data berhasil di ambil atau gagal.

```php
$response_data = json_decode($response, true);

if ($response_data && $response_data['error'] == 0) {
    foreach ($response_data['data'] as $produk) {
        $id_kategori = $this->Produk_model->get_or_insert_kategori($produk['kategori']);
        $id_status = $this->Produk_model->get_id_status($produk['status']);

        $data_produk = [
            'id_produk' => $produk['id_produk'],
            'nama_produk' => $produk['nama_produk'],
            'harga' => $produk['harga'],
            'id_kategori' => $id_kategori,
            'id_status' => $id_status
        ];

        $this->Produk_model->insert_produk($data_produk);
    }
    echo "Data berhasil dimasukkan ke dalam database!";
} else {
    echo "Terjadi kesalahan saat mengambil data dari API.";
}
```

Gambar 2.3 konfigurasi pengambilan data pada Produk.php

3. Konfigurasi models/Produk_model

File models bertujuan sebagai penghubung data antara website dengan database. Didalamnya terdapat beberapa fungsi untuk melakukan penampilan, pengambilan, perubahan, dan penghapusan data produk. Dengan ini, data di database dapat ditampilkan dan dikelola dengan mudah dari website.

```php
public function __construct() {
    parent::__construct();
    $this->load->database();
}

public function get_all_produk() {
    return $this->db->select('produk.*, kategori.nama_kategori, status.nama_status')
                    ->from('produk')
                    ->join('kategori', 'produk.id_kategori = kategori.id_kategori', 'left')
                    ->join('status', 'produk.id_status = status.id_status', 'left')
                    ->get()->result();
}

public function get_bisa_dijual() {
    return $this->db->select('produk.*, kategori.nama_kategori, status.nama_status')
                    ->from('produk')
                    ->join('kategori', 'produk.id_kategori = kategori.id_kategori', 'left')
                    ->join('status', 'produk.id_status = status.id_status', 'left')
                    ->where('status.nama_status', 'bisa dijual')
                    ->get()->result();
}
```

Gambar 3.1 fungsi menampilkan data dan filter data yang bisa dijual.

```php
public function get_or_insert_kategori($nama_kategori) {
    $this->db->where('nama_kategori', $nama_kategori);
    $query = $this->db->get('kategori');

    if ($query->num_rows() > 0) {
        return $query->row()->id_kategori;
    } else {
        $this->db->insert('kategori', ['nama_kategori' => $nama_kategori]);
        return $this->db->insert_id();
    }
}

public function get_kategori_by_name($nama_kategori) {
    return $this->db->get_where('kategori', ['nama_kategori' => $nama_kategori])->row();
}

public function insert_kategori($data) {
    $this->db->insert('kategori', $data);
    return $this->db->insert_id();
}
```

Gambar 3.2 fungsi mengatur data kategori

```php
public function get_status() {
    return $this->db->get('status')->result();
}

public function get_id_status($status) {
    return ($status == 'bisa dijual') ? 1 : 2;
}
```

Gambar 3.3 fungsi mengatur data status

```php
public function insert_produk($data) {
    return $this->db->insert('produk', $data);
}

public function get_produk_by_id($id) {
    return $this->db->get_where('produk', ['id_produk' => $id])->row();
}

public function update_produk($id, $data) {
    return $this->db->where('id_produk', $id)->update('produk', $data);
}

public function delete_produk($id) {
    return $this->db->delete('produk', ['id_produk' => $id]);
}
```

Gambar 3.4 fungsi memasukkan, mengubah, dan menghapus data.

4. Konfigurasi controllers/Produk.php

Seusai namanya, file controller ini bertujuan untuk mengatur sistem pada halaman website. Seperti pada perintah mengambil data API yang dibahas sebelumnya, file ini juga berfungsi mengatur system control seperti menampilkan data dan fungsi tombol pada halaman.

Pada halaman index diberikan perintah untuk mengambil semua data dari database dan menampilkannya sesuai dengan filter status yang aktif.

```php
public function index() {
    $filter_status = $this->input->get('status');

    $this->db->select('produk.*, kategori.nama_kategori, status.nama_status');
    $this->db->from('produk');
    $this->db->join('kategori', 'produk.id_kategori = kategori.id_kategori', 'left');
    $this->db->join('status', 'produk.id_status = status.id_status', 'left');

    if ($filter_status == '1') {
        $this->db->where('produk.id_status', 1);
    }

    $data['produk_list'] = $this->db->get()->result();
    $data['filter_status'] = $filter_status;

    $this->load->view('produk/index', $data);
}
```

Gambar 4.1 fungsi menampilkan data pada halaman index.php

Pada halaman tambah, terdapat fungsi form validation untuk memastikan bahwa data yang dimasukkan tidak boleh kosong dan sesuai dengan input yang diminta. Terdapat juga fungsi untuk menampilkan list kategori yang tersedia untuk memudahkan pengguna mengisi data. Selain itu kategori baru juga dapat ditambahkan jika diperlukan. Data yang sudah diisi kemudian dimasukkan ke dalam database

```php
public function tambah() {
    $this->form_validation->set_rules('nama_produk', 'Nama Produk', 'required');
    $this->form_validation->set_rules('harga', 'Harga', 'required|numeric');

    if ($this->form_validation->run() == FALSE) {
        $data['kategori_list'] = $this->db->get('kategori')->result();
        $this->load->view('produk/tambah', $data);
    } else {
        $id_kategori = $this->input->post('id_kategori');
        $nama_kategori = $this->input->post('nama_kategori');
        $id_status = $this->input->post('status');

        if ($id_kategori == "new" && !empty($nama_kategori)) {
            $kategori_data = ['nama_kategori' => $nama_kategori];
            $this->db->insert('kategori', $kategori_data);
            $id_kategori = $this->db->insert_id();
        }

        $data = [
            'nama_produk' => $this->input->post('nama_produk'),
            'harga' => $this->input->post('harga'),
            'id_kategori' => $id_kategori,
            'id_status' => $id_status
        ];

        $this->db->insert('produk', $data);
        redirect('produk');
    }
}
```

Gambar 4.2 fungsi halaman tambah.php

Pada halaman edit juga sama dengan halaman tambah, yang membedakan adalah tambahan perintah untuk mengambil data yang akan diedit berdasarkan id_produk. Halaman edit juga memiliki fungsi tambahan untuk mengupdate data yang telah diubah ke dalam database.

```php
public function edit($id) {
    $this->form_validation->set_rules('nama_produk', 'Nama Produk', 'required');
    $this->form_validation->set_rules('harga', 'Harga', 'required|numeric');

    $data['produk'] = $this->Produk_model->get_produk_by_id($id);
    $data['kategori_list'] = $this->db->get('kategori')->result();

    if ($this->form_validation->run() == FALSE) {
        $this->load->view('produk/edit', $data);
    } else {
        $id_kategori = $this->input->post('id_kategori');
        $nama_kategori_baru = trim($this->input->post('nama_kategori'));

        if ($id_kategori == 'new' && !empty($nama_kategori_baru)) {
            $this->db->insert('kategori', ['nama_kategori' => $nama_kategori_baru]);
            $id_kategori = $this->db->insert_id();
        }

        $update_data = [
            'nama_produk' => $this->input->post('nama_produk'),
            'harga' => $this->input->post('harga'),
            'id_kategori' => $id_kategori,
            'id_status' => $this->input->post('id_status')
        ];

        $this->Produk_model->update_produk($id, $update_data);
        redirect('produk');
    }
}
```

Gambar 4.3 fungsi halaman edit.php

```php
public function update($id_produk) {
    $this->load->library('form_validation');

    $this->form_validation->set_rules('nama_produk', 'Nama Produk', 'required');
    $this->form_validation->set_rules('harga', 'Harga', 'required|numeric');

    if ($this->form_validation->run() == FALSE) {
        $produk = $this->db->get_where('produk', ['id_produk' => $id_produk])->row();
        $kategori_list = $this->db->get('kategori')->result();
        $this->load->view('produk/edit', [
            'produk' => $produk,
            'kategori_list' => $kategori_list
        ]);
    } else {
        $id_kategori = $this->input->post('id_kategori');
        $nama_kategori_baru = trim($this->input->post('nama_kategori'));

        if ($id_kategori == 'new' && !empty($nama_kategori_baru)) {
            $this->db->insert('kategori', ['nama_kategori' => $nama_kategori_baru]);
            $id_kategori = $this->db->insert_id();
        }

        $data = [
            'nama_produk' => $this->input->post('nama_produk'),
            'harga' => $this->input->post('harga'),
            'id_kategori' => $id_kategori,
            'id_status' => $this->input->post('status')
        ];

        $this->db->update('produk', $data, ['id_produk' => $id_produk]);
        $this->session->set_flashdata('success', 'Produk berhasil diperbarui.');
        redirect('produk');
    }
}
```

Gambar 4.4 fungsi update data produk

Pada halaman produk terdapat tombol untuk menghapus data produk yang dipilih.

```php
public function hapus($id) {
    $this->Produk_model->delete_produk($id);
    redirect('produk');
}
```

Gambar 4.5 fungsi menghapus data produk

5. Konfigurasi halaman website.

Setelah semua file fungsi system website telah diatur, saatnya untuk mengatur tampilan setiap halaman website. Diawali dengan halaman produk sebagai halaman utama. Pada halaman ini, data produk yang ada dalam database ditampilkan. Data-data ini kemudian bisa difilter menggunakan tombol di kanan atas untuk menampilkan semua produk atau hanya produk yang bisa dijual. Terdapat tombol tambah produk di kiri atas dan tombol edit dan hapus pada setiap baris data. Jika tombol hapus ditekan maka muncul pop up notifikasi peringatan dan konfirmasi jika yakin ingin menghapus data.

Pada halaman tambah, terdapat inputan nama produk, harga, kategori dan status, nama produk dan harga wajib diisi, dan harga harus berupa angka. Inputan kategori dan status berupa dropdown yang memungkinkan pengguna untuk memilih data. khusus untuk kategori terdapat opsi untuk memasukkan kategori baru jika diperlukan.

Pada halaman edit, halaman akan menampilkan informasi dari data yang dipilih untuk diedit. Sistem halamannya mirip dengan halaman tambah produk dimana nama dan harga harus diisi dan harga harus berupa angka.

Semua halaman kemudian diberi pengaturan tampilan menggunakan file style.css agar tampilan lebih menarik dan enak dipandang. Saya mencoba mengambil insprasi dari kombinasi warna yang ada pada website resmi FastPrint.co.id dengan background putih dengan aksen kuning dan tulisan biru.