LAPORAN HASIL KODE PROGRAM CRUD USER PHP MySQL

Disusun guna memenuhi tugas mata kuliah Pemrograman Berbasis Platform

Disusun oleh:

23051204067 Raihan Rizki Alfareza

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SURABAYA

TAHUN 2024

DAFTAR ISI

1. Pendahuluan

Sistem CRUD (Create, Read, Update, Delete) adalah fondasi penting dalam pengembangan aplikasi yang berhubungan dengan basis data. Dalam laporan ini, saya akan membahas bagaimana cara mengimplementasikan sistem CRUD untuk mengelola data pengguna, menggunakan bahasa pemrograman PHP dan basis data MySQL.

Aplikasi ini dirancang untuk memiliki dua role pengguna: admin dan user. Admin memiliki akses penuh untuk mengelola data pengguna, sedangkan user hanya bisa melihat dan mengedit data miliknya sendiri.

2. Struktur Aplikasi

Struktur direktori aplikasi terdiri dari beberapa folder utama sebagai berikut:

```
├── admin/      Berisi file terkait fungsi admin
├── user/       Berisi file terkait fungsi user
├── layout/     Berisi file template tata letak (header dan footer)
├── services/   Berisi file koneksi database
├── assets/     Berisi file aset (CSS, JavaScript, gambar)
└── index.php   Halaman utama aplikasi (login/register)
```

3. Koneksi Database

Koneksi database merupakan komponen penting dalam aplikasi CRUD. File services/database.php bertugas menghubungkan aplikasi dengan database MySQL.

```php
<?php
$hostname = "localhost";
$username = "root";
$password = "";
$database = "informatika";

$login = mysqli_connect($hostname, $username, $password, $database);

echo "Berhasil terkoneksi database";
?>
```

Penjelasan:

- Variabel $hostname, $username, $password, dan $database menyimpan informasi koneksi ke database yang disimpan pada variabel $login untuk dipanggil dalam fungsi yang membutuhkan fetch data dari database.

- Fungsi mysqli_connect() digunakan untuk membuat koneksi ke database MySQL.

- Pesan “Berhasil terkoneksi database” ditampilkan jika koneksi berhasil.

4. **Fungsi Utama**

Fungsi-fungsi utama untuk operasi database didefinisikan dalam file admin/functions.php dan user/functions.php. Fungsi-fungsi ini akan digunakan di seluruh aplikasi untuk interaksi dengan database sesuai dengan role usernya.

```php
<?php
// Mengambil banyak data
function ambildata($login, $query) {
    $data = mysqli_query($login, $query);
    if (mysqli_num_rows($data) > 0) {
        while($row = mysqli_fetch_assoc($data)) {
            $hasil[] = $row;
        }
        return $hasil;
    }
}

// Menjalankan query (insert, update, delete)
function bisa($login, $query) {
    $db = mysqli_query($login, $query);
    if($db) {
        return 1;
    } else {
        return 0;
    }
}

// Mengambil satu baris data
function ambilsatubaris($login, $query) {
    $db = mysqli_query($login, $query);
    return mysqli_fetch_assoc($db);
}
?>
```

Penjelasan:

- Fungsi ambildata() mengambil banyak data dari database dan mengembalikannya dalam bentuk array.

- Fungsi bisa() menjalankan query (insert, update, delete) dan mengembalikan nilai 1 jika berhasil atau 0 jika gagal. Ini berfungsi untuk push notifikasi status keberhasilan fitur.

- Fungsi ambilsatubaris() mengambil satu baris data dari database.

5. Sistem Autentikasi

5.1. Login

Sistem login diimplementasikan dalam file index.php dan ceklogin.php. Proses login dimulai dari form pada index.php:

```
<form class="login100-form validate-form" method="POST" action="ceklogin.php">
    <div class="wrap-input100 validate-input m-b-26" data-validate="Username is
required">
      <span class="label-input100">Username</span>
      <input class="input100" type="text" name="username" placeholder="Enter username">
      <span class="focus-input100"></span>
  </div>

  <div class="wrap-input100 validate-input m-b-18" data-validate = "Password is
required">
      <span class="label-input100">Password</span>
      <input class="input100" type="password" name="password" placeholder="Enter
password">
      <span class="focus-input100"></span>
  </div>
  <?php if (isset($_GET['msg'])): ?>
      <small class="text-danger"><?= $_GET['msg']; ?></small>
  <?php endif ?>
  <div class="container-login100-form-btn">
      <button class="login100-form-btn" type="submit">
        Login
      </button>
  </div>
</form>
```

Kemudian data form dikirim ke ceklogin.php untuk diproses:

```
<?php
session_start();
include "services/database.php";

$username = ($_POST['username']);
$password = md5($_POST['password']);
$query = "SELECT * FROM users where username='$username' AND password =
'$password'";
$row = mysqli_query($login, $query);
$data = mysqli_fetch_assoc($row);
```

```php
$cek = mysqli_num_rows($row);

if($cek > 0) {
   if($data['role'] == 'admin') {
      $_SESSION['role'] = 'admin';
      $_SESSION['username'] = $data['username'];
      $_SESSION['password'] = $data['password'];
      $_SESSION['user_id'] = $data['id'];
      header('Location: /teknik_informatika/admin/pengguna.php');
      exit;
   } else if($data['role'] == 'user') {
      $_SESSION['role'] = 'user';
      $_SESSION['username'] = $data['username'];
      $_SESSION['password'] = $data['password'];
      $_SESSION['user_id'] = $data['id'];
      header('Location: /teknik_informatika/user/pengguna.php');
      exit;
   }
} else {
   $msg = 'Username Atau Password Anda Salah';
   header('location:index.php?msg='.$msg);
}
?>
```

Penjelasan:

- session_start() digunakan untuk memulai session.
- Username dan password (setelah dienkripsi dengan md5) diambil dari form login.
- Query SQL dijalankan untuk mencari pengguna dengan username dan password yang sesuai.
- Jika ditemukan, sistem akan memverifikasi role pengguna (admin atau user) dan diarahkan ke halaman yang sesuai.
- Jika tidak ditemukan, akan mengembalikan ke halaman login dengan pesan error.

5.2. Registrasi

Sistem registrasi diimplementasikan dalam file user/register.php. Pengguna dapat mendaftar sebagai user baru melalui form registrasi:

```php
<?php
include "../services/database.php";

$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $nama     = mysqli_real_escape_string($login, $_POST['nama']);
    $username = mysqli_real_escape_string($login, $_POST['username']);
    $email    = mysqli_real_escape_string($login, $_POST['email']);
    $password = md5(mysqli_real_escape_string($login, $_POST['password']));

    $cek = "SELECT id FROM users WHERE username='$username'";
    if (mysqli_num_rows(mysqli_query($login, $cek)) > 0) {
        $error = 'Username sudah terpakai';
    } else {
        $ins = "
          INSERT INTO users
            (nama_lengkap, username, email, password, role)
          VALUES
            ('$nama','$username','$email','$password','user')
        ";
        if (mysqli_query($login, $ins)) {
            header('Location: ../index.php?msg=Registrasi berhasil, silakan login');
            exit;
        } else {
            $error = 'Error: ' . mysqli_error($login);
        }
    }
}
?>
```

Penjelasan:

- Fungsi mysqli_real_escape_string() digunakan untuk mencegah SQL Injection.
- Sistem memeriksa apakah username sudah terpakai.
- Password dienkripsi menggunakan fungsi md5() sebelum disimpan ke database.
- Jika registrasi berhasil, pengguna diarahkan ke halaman login dengan default role user (karena fitur register hanya untuk pengguna baru dengan role user).

5.3. Logout

Mekanisme logout diimplementasikan dalam file admin/logout.php dan user/logout.php:

```php
<?php
session_start();
session_destroy();
header('location:../index.php');
?>
```

Penjelasan:

- session_start() digunakan untuk memulai session.
- session_destroy() menghapus semua data session.
- Setelah logout, pengguna diarahkan kembali ke halaman login.

6. Operasi CRUD (Admin)

Admin memiliki akses penuh untuk mengelola data pengguna melalui fitur CRUD.

6.1. Read (Membaca Data)

Implementasi untuk membaca data pengguna terdapat dalam file admin/pengguna.php:

```php
<?php
$title = 'pengguna';
require 'functions.php';
require '../layout/layout_header.php';
$query = 'SELECT * FROM `users`';
$data = ambildata($login, $query);
?>
```

Kemudian data ditampilkan dalam tabel:

```
<tbody>
    <?php foreach($data as $user): ?>
            <tr>
        <td></td>
        <td><?= $user['username'] ?></td>
        <td><?= $user['nama_lengkap'] ?></td>
        <td><?= $user['email'] ?></td>
        <td align="center">
          <a href="pengguna_edit.php?id=<?php echo $user['id']; ?>" class="btn btn-
success btn-sm">
            <i class="fa fa-edit"></i> Edit
        </a>
         
```

```
            <a href="pengguna_hapus.php?id=<?php echo $user['id']; ?>" class="btn btn-
danger btn-sm">
              <i class="fa fa-trash"></i> Hapus
            </a>
          </td>
        </tr>
    <?php endforeach; ?>
</tbody>
```

6.2. Create (Menambah Data)

Implementasi untuk menambah data pengguna terdapat dalam file admin/pengguna_tambah.php:

```
<?php
$title = 'pengguna';
require 'functions.php';
$roles = ['admin','user'];

if (isset($_POST['btn-simpan'])) {
    $nama     = $_POST['nama_lengkap'];
    $username = $_POST['username'];
    $pass     = md5($_POST['password']);
    $email    = $_POST['email'];
    $role     = $_POST['role'];

    $query = "
      INSERT INTO users
        (nama_lengkap, username, password, email, role)
      VALUES
        ('$nama', '$username', '$pass', '$email', '$role')
    ";

    if (bisa($login, $query) == 1) {
        header('Location: pengguna.php?crud=true&msg=Berhasil menambahkan '.$role);
        exit;
    } else {
        echo "<div class='alert alert-danger'>Gagal menambahkan data</div>";
    }
}
?>
```

Form untuk menambah data:

```
<form method="post">
  <div class="form-group">
```

```
    <label>Nama Lengkap</label>
    <input type="text" name="nama_lengkap" class="form-control" required>
  </div>
  <div class="form-group">
    <label>Username</label>
    <input type="text" name="username" class="form-control" required>
  </div>
  <div class="form-group">
    <label>Password</label>
    <input type="password" name="password" class="form-control" required>
  </div>
  <div class="form-group">
    <label>Email</label>
    <input type="text" name="email" class="form-control" required>
  </div>
  <div class="form-group">
    <label>Role</label>
    <select name="role" class="form-control" required>
      <?php foreach($roles as $r): ?>
        <option value="<?= $r ?>"><?= ucfirst($r) ?></option>
      <?php endforeach; ?>
    </select>
  </div>
  <button type="submit" name="btn-simpan" class="btn btn-primary">Simpan</button>
  <a href="pengguna.php" class="btn btn-default">Batal</a>
</form>
```

6.3. Update (Mengubah Data)

Implementasi untuk mengubah data pengguna terdapat dalam file admin/pengguna_edit.php:

```
<?php
$title = 'pengguna';
require 'functions.php';

$role = ['admin','user'];

$id_user = $_GET['id'];
$queryedit = "SELECT * FROM users WHERE id = '$id_user'";
$edit = ambilsatubaris($login, $queryedit);

if(isset($_POST['btn-simpan'])) {
    $nama = $_POST['nama_lengkap'];
    $username = $_POST['username'];
    $email = $_POST['email'];
    $role = $_POST['role'];
```

```php
    $passwordlam = $_POST['passlama'];
    $querypass = "SELECT * FROM users WHERE id = '$id_user'";
    $res = mysqli_query($login, $querypass);
    $old = mysqli_fetch_assoc($res);
    $email_lama = $old['email'];

    if (md5($_POST['passlama']) !== $old['password']) {
        echo "<script>alert('Password lama tidak
sesuai');location.href='pengguna_edit.php?id=$id_user';</script>";
        exit;
    }
    if (!empty($_POST['password'])) {
        $passwordToSave = md5(mysqli_real_escape_string($login, $_POST['password']));
    } else {
        $passwordToSave = $old['password'];
    }

    if(!empty(trim($_POST['email']))) {
        $emailToSave = $email;
    } else {
        $emailToSave = $email_lama;
    }

    $query = "
      UPDATE users SET
        nama_lengkap = '$nama',
        username = '$username',
        email = '$emailToSave',
        role = '$role',
        password = '$passwordToSave'
      WHERE id = '$id_user'
    ";

    $execute = bisa($login, $query);
    if($execute == 1) {
        $success = 'true';
        $title = 'Berhasil';
        $message = 'Berhasil mengubah ' .$role;
        $type = 'success';
        header('location:
pengguna.php?crud='.$success.'&msg='.$message.'&type='.$type.'&title='.$title);
    } else {
        echo "Gagal Tambah Data";
    }
}
?>
```

Fitur penting dalam form edit:

- User harus verifikasi password lama sebelum melakukan perubahan untuk keamanan tambahan.
- Opsi untuk mengubah password (opsional).
- Mempertahankan password lama secara otomatis jika tidak ada perubahan (field password baru dikosongkan).

6.4. Delete (Menghapus Data)

Implementasi untuk menghapus data pengguna terdapat dalam file admin/pengguna_hapus.php:

```
<?php
require 'functions.php';

if (!isset($_GET['id'])) {
    echo "<script>alert('ID Pengguna tidak
ditemukan!');location.href='pengguna.php';</script>";
    exit;
}
$id = $_GET['id'];

if (isset($_POST['btn-simpan'])) {
    $input = md5($_POST['passlama']);

    $adminId   = $_SESSION['user_id'];
    $querypass = "SELECT password FROM users WHERE role = 'admin' AND id =
'{$adminId}' LIMIT 1";
    $res = mysqli_query($login, $querypass);
    $admin = mysqli_fetch_assoc($res);

    if (($input) !== $admin['password']) {
        echo "<script>alert('Password tidak
sesuai');location.href='pengguna_hapus.php?id={$id}';</script>";
        exit;
    }

    $sql = "DELETE FROM users WHERE id = {$id}";
    if (mysqli_query($login, $sql)) {
        header("location: pengguna.php?crud=true&msg=Pengguna dihapus&type=success");
        exit;
    } else {
        echo "<script>alert('Gagal menghapus:
```

```
".mysqli_error($login).'');location.href='pengguna.php';</script>";
        exit;
    }
}
?>
```

Fungsi penghapusan data memiliki fitur keamanan tambahan yang mewajibkan admin memasukkan password untuk konfirmasi penghapusan:

```
<form method="post" action="">
    <div class="form-group">
        <label>Masukkan Password Admin untuk Menghapus!</label>
        <input type="password" name="passlama" required class="form-control">
    </div>
    <div class="text-right">
        <button type="reset" class="btn btn-danger">Reset</button>
        <button type="submit" name="btn-simpan" class="btn btn-primary">Hapus</button>
    </div>
</form>
```

7. Operasi CRUD (User)

Berbeda dengan admin, user hanya memiliki akses terbatas terhadap data. User hanya dapat melihat dan mengedit data pribadinya.

7.1. Read (Membaca Data)

Implementasi untuk membaca data user terdapat dalam file user/pengguna.php:

```
<?php
session_start();
$title = 'pengguna';
require 'functions.php';
require '../layout/layout_header.php';
$userId = $_SESSION['user_id'];
$query = "SELECT * FROM users WHERE id = '$userId'";
$data = ambildata($login, $query);
?>
```

Data hanya ditampilkan untuk pengguna yang sedang login (data diambil dari id pengguna melalui session):

```
<tbody>
    <?php if (!empty($data)):
        $u = $data[0]; ?>
    <tr>
```

```
        <td>1</td>
        <td><?= htmlspecialchars($u['username']) ?></td>
        <td><?= htmlspecialchars($u['nama_lengkap']) ?></td>
        <td><?= htmlspecialchars($u['email']) ?></td>
        <td align="center">
        <a href="pengguna_edit.php?id=<?= $u['id'] ?>"
           class="btn btn-success btn-sm">
           <i class="fa fa-edit"></i> Edit
        </a>
        </td>
    </tr>
    <?php else: ?>
    <tr><td colspan="5" class="text-center">Data tidak ditemukan</td></tr>
    <?php endif; ?>
</tbody>
```

Saya menambahkan penggunaan htmlspecialchars() untuk mencegah XSS (Cross-Site Scripting).

7.2. Update (Mengubah Data)

Implementasi untuk mengubah data user terdapat dalam file user/pengguna_edit.php:

```
<?php
$title = 'pengguna';
require 'functions.php';

$role = ['user'];

$id_user = $_GET['id'];
$queryedit = "SELECT * FROM users WHERE id = '$id_user'";
$edit = ambilsatubaris($login, $queryedit);

if(isset($_POST['btn-simpan'])) {
    $nama     = $_POST['nama_lengkap'];
    $username = $_POST['username'];
    $email = $_POST['email'];
    $role     = $_POST['role'];
    $passwordlam = $_POST['passlama'];
    $querypass = "SELECT password FROM users WHERE id = '$id_user'";
    $res = mysqli_query($login, $querypass);
    $old = mysqli_fetch_assoc($res);
    $email_lama = $old['email'];

    if (md5($_POST['passlama']) !== $old['password']) {
        echo "<script>alert('Password lama tidak
```

```
sesuai');location.href='pengguna_edit.php?id=$id_user';</script>";
        exit;
    }
    if (!empty($_POST['password'])) {
        $passwordToSave = ($_POST['password']);
    } else {
        $passwordToSave = $old['password'];
    }
    if (!empty($_POST['email'])) {
        $emailToSave = $email;
    } else {
        $emailToSave = $email_lama;
    }

    $query = "
      UPDATE users SET
        nama_lengkap = '$nama',
        username = '$username',
        email = '$email',
        password = '$passwordToSave'
      WHERE id = '$id_user'
    ";

    $execute = bisa($login, $query);
    if($execute == 1) {
        $success = 'true';
        $title = 'Berhasil';
        $message = 'Berhasil mengubah ' .$role;
        $type = 'success';
        header('location:
pengguna.php?crud='.$success.'&msg='.$message.'&type='.$type.'&title='.$title);
    } else {
        echo "Gagal Tambah Data";
    }
}
?>
```

Perbedaan utama dengan admin:

- User tidak dapat mengubah role (default value edit adalah role user yang di hard code pada backend, tidak ditampilkan pada frontend).

- User hanya dapat mengedit data miliknya sendiri berdasarkan session id user terkait.

8. Struktur Database

Struktur tabel users yang digunakan dalam aplikasi ini:

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `nama_lengkap` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `role` enum('admin','user') DEFAULT 'user',
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Penjelasan struktur tabel:

- id: Primary key dengan auto increment.
- username: Username pengguna (unik).
- password: Password pengguna (dienkripsi dengan md5 agar saya dapat mendecryptnya dengan mudah bila lupa password).
- nama_lengkap: Nama lengkap pengguna.
- email: Alamat email pengguna.
- role: Peran pengguna (admin atau user).
- created_at: Waktu pembuatan akun (otomatis diisi).

9. Keamanan Aplikasi

Aplikasi ini mengimplementasikan beberapa aspek keamanan yang telah diberikan pada materi kuliah sebelumnya:

1. **Validasi Input**: Memastikan input pengguna sesuai dengan yang diharapkan.
2. **Verifikasi Password**: Memastikan pengguna memasukkan password lama sebelum melakukan perubahan data.
3. **Enkripsi Password**: Password dienkripsi menggunakan fungsi md5 sebelum disimpan ke database.

4. **Session Management**: Kontrol akses berbasis session untuk membedakan pengguna admin dan user.
5. **Pencegahan XSS**: Penggunaan htmlspecialchars() untuk mencegah Cross-Site Scripting.

10. Tata Letak dan Tampilan

Aplikasi menggunakan struktur tata letak yang konsisten dengan template header dan footer yang sama untuk semua halaman. File layout/layout_header.php dan layout/layout_footer.php berisi kode HTML yang digunakan di seluruh aplikasi.

Tampilan aplikasi menggunakan framework CSS Bootstrap dan beberapa plugin tambahan untuk meningkatkan tampilan visual (kebetulan saya menggunakan bekas source code ketika SMK).

11. Alur Kerja Aplikasi

1. **Login/Register**:
    - Pengguna mengakses index.php untuk login.
    - Jika belum memiliki akun, pengguna dapat mendaftar melalui user/register.php.
    - Setelah login, pengguna diarahkan ke halaman sesuai role yang ada (admin atau user).
2. **Admin**:
    - Admin dapat melihat semua data pengguna di admin/pengguna.php.
    - Admin dapat menambahkan pengguna baru di admin/pengguna_tambah.php.
    - Admin dapat mengedit data pengguna di admin/pengguna_edit.php.
    - Admin dapat menghapus pengguna di admin/pengguna_hapus.php.
3. **User**:
    - User hanya dapat melihat data pribadinya di user/pengguna.php.

- User dapat mengedit data pribadinya di user/pengguna_edit.php.

4. **Logout**:

    - Pengguna dapat logout melalui menu yang tersedia.
    - Setelah logout, session dihapus dan pengguna diarahkan kembali ke halaman login.

12. Kesimpulan

Aplikasi CRUD User sederhana ini berhasil mengimplementasikan operasi dasar database (Create, Read, Update, Delete) dengan sistem autentikasi dan otorisasi berbasis role user session. Aplikasi ini dapat digunakan sebagai dasar untuk pengembangan aplikasi web berbasis PHP dan MySQL yang lebih kompleks kedepannya sesuai dengan mata kuliah PBP yang sedang berjalan.