LAPORAN
PEMOGRAMAN WEB
PENERAPAN KONSEP MVC (*MODEL VIEW-CONTROLLER*)

Dosen Pengampu: Berta Erwin Slam, S.T., M.Kom

Disusun oleh:

Rodiyan Ramadhani	2301020063
Widuri Eka Febriyanti	2301020017
Adelia Kristianti Purba	2301020105
Hermawan	2301020025
Imelda Cristina	2301020031

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN
UNIVERSITAS MARITIM RAJA ALI HAJI
2025

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Model–View–Controller (MVC) merupakan pola arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga komponen utama: *Model* untuk mengelola data, *View* untuk menampilkan antarmuka, dan *Controller* untuk mengatur alur logika aplikasi. Pemisahan ini membuat sistem lebih terstruktur, mudah dikembangkan, dan lebih sederhana dalam pemeliharaan.

Penerapan MVC sangat relevan pada pengembangan aplikasi berbasis web. Dengan memanfaatkan MVC, pengelolaan data dan tampilan dapat diatur secara terpisah sehingga perubahan pada salah satunya tidak akan memengaruhi bagian lain. Oleh karena itu, penerapan konsep MVC penting untuk dipelajari dan diimplementasikan dalam praktik pengembangan aplikasi.

1.2 Rumusan Masalah

1. Bagaimana penerapan konsep MVC pada pengembangan aplikasi berbasis web?
2. Bagaimana interaksi antara *Model*, *View*, dan *Controller* dalam aplikasi?

1.3 Tujuan Proyek

1. Menerapkan pola MVC menggunakan PHP dan MySQL.
2. Menjelaskan penerapan konsep MVC pada aplikasi berbasis web.

BAB II
HASIL DAN PEMBAHASAN

2.1 Pembagian Kerja

Pembagian kerja tim dijelaskan menggunakan **tabel** sebagai berikut:

Table 1 Tabel Pembagian Kerja Tim

No	Nama	Posisi Tim	Deskripsi Kerja
1	Rodiyan Ramadhani	Ketua	Project Manager, membuat program untuk folder views, finishing
2	Adelia Kristianti Purba	Anggota	Membuat program untuk folder models, laporan
3	Widuri Eka Febriyanti	Anggota	Membuat program Config, database
4	Hermawan	Anggota	Membuat program untuk folder controller
5	Imelda	Anggota	Membuat program index

2.2 Hasil dan Pembahasan Proyek

Tugas ini bertujuan untuk menerapkan pola arsitektur Model–View–Controller (MVC) dalam pengembangan aplikasi berbasis web menggunakan bahasa pemrograman PHP dan database MySQL. Menggunakan sebuah database bernama db_mahasiswa yang berfungsi sebagai tempat penyimpanan data mahasiswa. Di dalam database ini terdapat sebuah tabel utama yaitu mahasiswa. Tabel ini menyimpan informasi dasar setiap mahasiswa dengan struktur sebagai berikut:

- id → kolom bertipe *integer* yang berfungsi sebagai *primary key* dan dibuat otomatis bertambah (*auto increment*). Kolom ini digunakan sebagai identitas unik untuk membedakan setiap data mahasiswa.
- nim → kolom bertipe *varchar(20)* yang menyimpan Nomor Induk Mahasiswa. Kolom ini menjadi atribut penting karena setiap mahasiswa pasti memiliki NIM yang berbeda.
- nama → kolom bertipe *varchar(100)* yang menyimpan nama lengkap mahasiswa.

- jurusan → kolom bertipe *varchar(50)* yang menyimpan informasi jurusan mahasiswa, misalnya “Teknik Informatika” atau “Sistem Informasi”.

Aplikasi yang dibuat berupa sistem sederhana untuk mengelola data mahasiswa dengan fitur CRUD (*Create, Read, Update, Delete*). Dalam implementasinya, aplikasi dipisahkan ke dalam tiga komponen utama, yaitu:

1. **Model**, berfungsi mengatur interaksi dengan database, termasuk menyimpan, mengambil, memperbarui, dan menghapus data mahasiswa. Pada program CRUD Mahasiswa ini, kita menggunakan konsep MVC (*Model–View–Controller*) agar struktur kodenya lebih rapi dan jelas fungsinya. Bagian Model bisa dilihat pada file Mahasiswa.php di folder models. Model ini bertugas untuk berhubungan langsung dengan database. Misalnya, saat kita ingin menampilkan semua data mahasiswa, kodenya seperti ini:

```
public function getAll() {
    $result = $this->conn->query("SELECT * FROM
mahasiswa");
    return $result->fetch_all(MYSQLI_ASSOC);
}
```

Kode di atas artinya Model melakukan query ke tabel mahasiswa, lalu mengambil semua datanya untuk dikirim ke bagian lain. Jadi, model ini ibarat "jembatan" ke database.

2. View, berfungsi menampilkan antarmuka kepada pengguna berupa form input data mahasiswa serta tampilan daftar mahasiswa. bagian View yang berfungsi untuk menampilkan antarmuka kepada pengguna. Contohnya ada di file views/mahasiswa/mahasiswa_list.php. Pada file ini kita membuat tampilan tabel untuk daftar mahasiswa, seperti berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>Daftar Mahasiswa</title>
    <link rel="stylesheet" href="views/style.css">
```

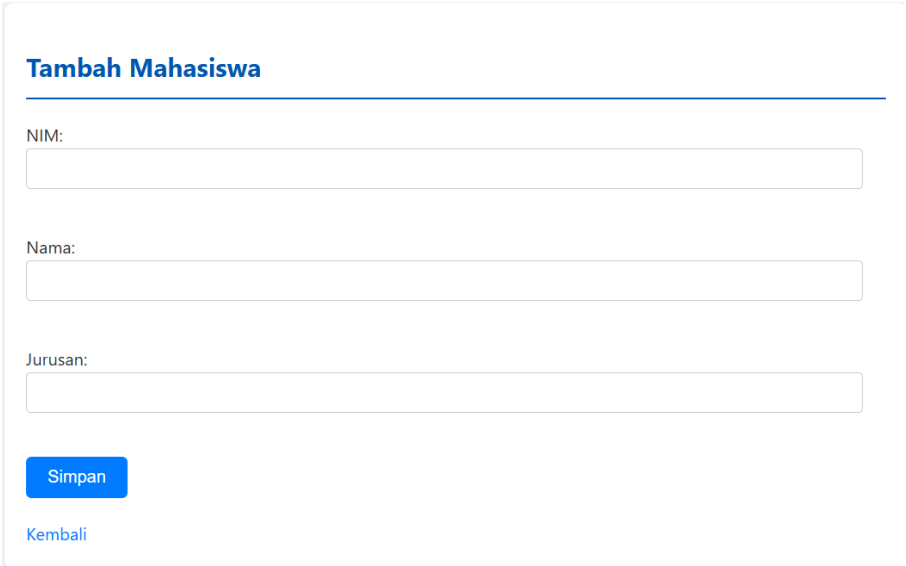
```
</head>
<body>
    <div class="container">
        <h2>Daftar Mahasiswa</h2>
        <a  href="index.php?action=create"  class="add-link">+  Tambah
Mahasiswa</a>
        <table>
            <tr>
                <th>No</th>
                <th>ID</th>
                <th>NIM</th>
                <th>Nama</th>
                <th>Jurusan</th>
                <th>Aksi</th>
            </tr>
            <?php
            $no = 1;
            while ($row = $result->fetch_assoc()) { ?>
            <tr>
                <td><?= $no++; ?></td>
                <td><?= $row['id']; ?></td>
                <td><?= $row['nim']; ?></td>
                <td><?= $row['nama']; ?></td>
                <td><?= $row['jurusan']; ?></td>
                <td class="action-links">
<a href="index.php?action=edit&id=<?= $row['id']; ?>">Edit</a> |
<a href="index.php?action=delete&id=<?= $row['id']; ?>"
onclick="return confirm('Yakin hapus?')">Hapus</a>
   </td>
  </tr>
            <?php } ?>
        </table>
    </div>
```

```
</body>
</html>
```

Di sinilah data yang sudah diambil dari database melalui Model, dan diteruskan oleh Controller, akhirnya ditampilkan dalam bentuk tabel agar bisa dilihat dan diakses pengguna. Selain menampilkan data, pada bagian Views juga disediakan halaman untuk menambahkan data baru, yaitu melalui form input. Kode program berikut adalah contoh halaman *Tambah Mahasiswa* yang memungkinkan pengguna memasukkan NIM, Nama, dan Jurusan:

```
            <form method="POST">
                NIM: <input type="text" name="nim" required>
                Nama: <input type="text" name="nama" required>
                Jurusan: <input type="text" name="jurusan" required>
                <button type="submit">Simpan</button>
            </form>
```

Form ini berfungsi untuk mengirim data mahasiswa baru ke sistem agar dapat disimpan ke dalam database melalui proses yang ditangani oleh Controller dan Model.

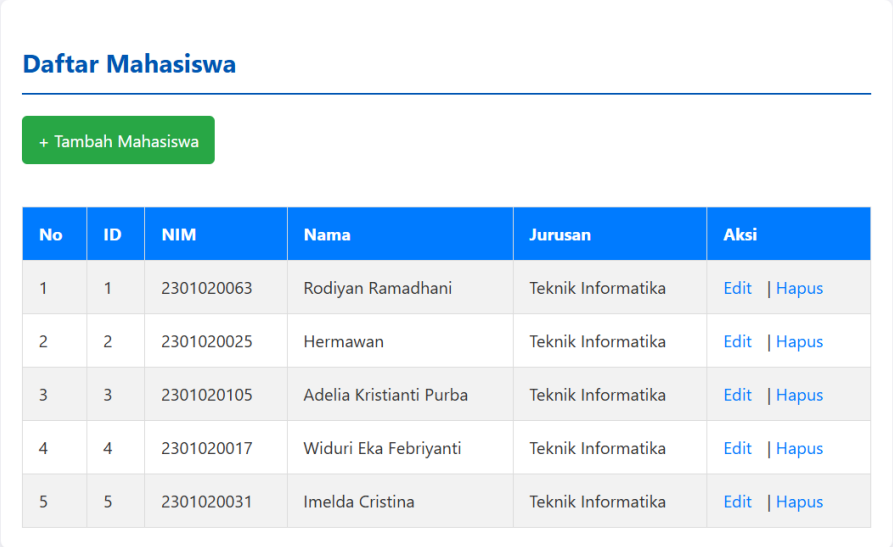


Gambar 1. Form Pengisian Data Mahasiswa

Pada bagian View, file mahasiswa_list.php digunakan untuk menampilkan seluruh data mahasiswa dalam bentuk tabel. Data yang ditampilkan di sini berasal dari database, yang sebelumnya sudah diambil oleh Model (Mahasiswa.php) dan diteruskan oleh Controller (MahasiswaController.php). Kodenya sebagai berikut:

```
while ($row = $result->fetch_assoc()) { ?>
            <tr>
                <td><?= $no++; ?></td>
                <td><?= $row['id']; ?></td>
                <td><?= $row['nim']; ?></td>
                <td><?= $row['nama']; ?></td>
                <td><?= $row['jurusan']; ?></td>
                <td class="action-links">
                    <a     href="index.php?action=edit&id=<?=     $row['id'];
?>">Edit</a> |
                    <a href="index.php?action=delete&id=<?= $row['id']; ?>"
                       onclick="return confirm('Yakin hapus?')">Hapus</a>
                </td>
            </tr>
            <?php } ?>
        </table>
    </div>
</body>
</html>
```

Selain menampilkan data, pada tabel daftar mahasiswa juga disediakan menu Edit dan Hapus pada kolom *Aksi*. Saat pengguna mengklik tautan Edit, maka sistem akan membaca id mahasiswa dari URL (`misalnya index.php?action=edit&id=2`). Nilai id ini diteruskan ke Controller (`MahasiswaController.php`) yang kemudian memanggil fungsi `getById($id)` dari Model untuk mengambil data mahasiswa sesuai id tersebut. Gambar berikut menunjukkan hasil tampilan daftar mahasiswa dan fitur edit:



Gambar 2. List mahasiswa dan Fitur Aksi

3. **Controller**, berfungsi menghubungkan antara Model dan View, mengatur alur logika aplikasi berdasarkan aksi yang dilakukan pengguna. Controller yang ada di file MahasiswaController.php, ini berperan sebagai penghubung antara Model dengan View. Misalnya, saat kita ingin menampilkan daftar mahasiswa, Controller akan memanggil data dari Model lalu mengirimkannya ke View. Contohnya bisa dilihat pada kode berikut:

```
public function index() {
      $result = $this->model->getAll();
      include "views/mahasiswa_list.php";
```

Dari potongan kode tersebut, Controller memanggil fungsi getAll() dari Model untuk mengambil data mahasiswa. Setelah itu, Controller memasukkan data tersebut ke file index.php yang ada di folder View, supaya bisa ditampilkan di layar.

Selain tiga bagian utama tadi, aplikasi ini juga memiliki dua file penting lain yang mendukung prinsip MVC, yaitu config.php dan index.php.

1. File config.php

File config.php berperan untuk menyiapkan koneksi ke database, sehingga semua proses pengambilan maupun penyimpanan data dapat berjalan dengan baik.

```
class DatabaseConnection {
    private $host = "localhost";
    private $username = "root";
    private $password = "";
    private $database = "db_mahasiswa";
    public $connection;

    public function __construct() {
        $this->connection = new mysqli($this->host,
$this->username, $this->password, $this->database);

        if ($this->connection->connect_error) {
            die("Koneksi gagal: " . $this->connection-
>connect_error);
        }
    }
}
```

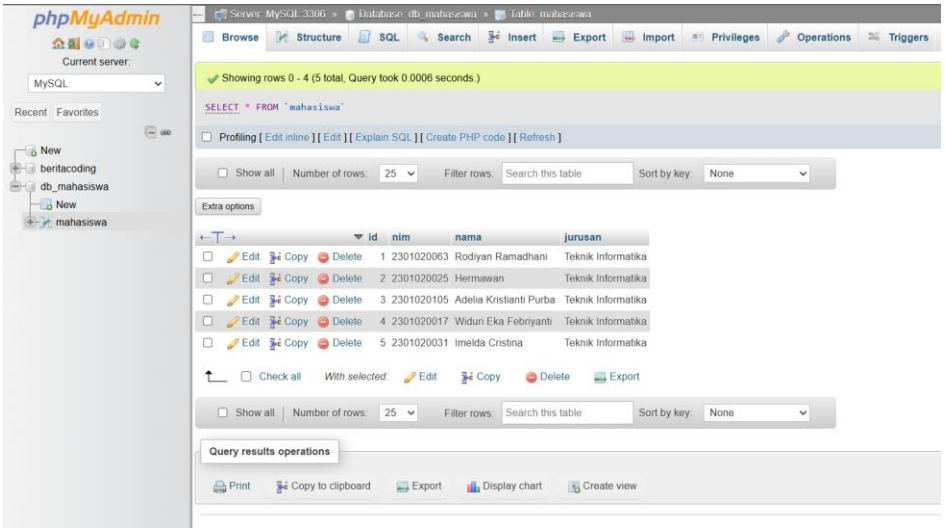
Bagian ini membuat kelas DatabaseConnection yang akan otomatis melakukan koneksi ke database MySQL ketika objeknya dibuat.

- Variabel $host, $username, $password, dan $database menyimpan informasi untuk login ke database.
- Konstruktor `__construct()` akan memanggil new mysqli(...) untuk membuka koneksi.
- Kalau koneksi gagal, muncul pesan *"Koneksi gagal"*.

Di akhir file, ada kode:

```
$db = new DatabaseConnection();
$conn = $db->connection;
```

Artinya objek DatabaseConnection langsung dibuat, lalu hasil koneksi disimpan di variabel `$conn`. Variabel $conn inilah yang akan dipakai oleh bagian lain (Controller dan Model) untuk mengakses database.



Gambar 3. Konfigurasi Database *phpmyAdmin*

Gambar di atas menunjukkan isi tabel mahasiswa yang sudah dibuat di dalam database. Tabel inilah yang menjadi pusat penyimpanan data, dan nantinya akan diakses melalui Model, dikirim oleh Controller, serta ditampilkan lewat View.

2. Index.php

File ini bisa disebut sebagai “pintu utama” aplikasi. Jadi ketika pengguna membuka aplikasi lewat browser, pertama kali yang dipanggil adalah index.php.

```
<?php
include "config.php";
include "controllers/MahasiswaController.php";

$controller = new MahasiswaController($conn);

$action = $_GET['action'] ?? 'index';
$id = $_GET['id'] ?? null;

switch ($action) {
```

```
        case 'create':
            $controller->create();
            break;
        case 'edit':
            $controller->edit($id);
            break;
        case 'delete':
            $controller->delete($id);
            break;
        default:
            $controller->index();
            break;
    }
    ?>
```

Penjelasan:

1. Baris include "`config.php`"; → supaya koneksi database tersedia.
2. Baris include "`controllers/MahasiswaController.php`"; → memanggil file Controller agar bisa digunakan.
3. `$controller = new MahasiswaController($conn);` → membuat objek Controller dengan membawa koneksi database (`$conn`) dari config.php.
4. Variabel $action → membaca parameter action dari URL, misalnya `index.php?action=create`. Kalau tidak ada parameter, otomatis nilainya 'index'.
5. `switch ($action)` → logika utama aplikasi. Bergantung pada nilai action, Controller akan menjalankan fungsi:
    - `create()` → untuk menambah data mahasiswa,
    - `edit($id)` → untuk mengedit data,
    - `delete($id)` → untuk menghapus data,
    - `index()` → default, menampilkan daftar mahasiswa.

    Jadi, index.php inilah yang mengatur alur aplikasi berdasarkan interaksi pengguna.

BAB III
KESIMPULAN DAN SARAN

3.1 Kesimpulan

Dari hasil pengerjaan tugas ini dapat disimpulkan bahwa penerapan konsep *Model–View–Controller* (MVC) telah berhasil dan sangat membantu dalam pengembangan aplikasi web yang lebih terstruktur dan mudah dipelihara. Dengan membagi kode ke dalam tiga bagian utama, yaitu Model, View, dan Controller, alur program menjadi lebih jelas:

1. Model mengatur interaksi dengan database (id, nim, nama, dan jurusan).
2. View bertanggung jawab menampilkan data kepada pengguna.
3. Controller menjadi penghubung yang mengatur logika alur antara model dan view.

Aplikasi yang dibuat berhasil menampilkan fitur CRUD (*Create, Read, Update, Delete*) untuk data mahasiswa yang meliputi *id, NIM, nama, dan jurusan*. Dengan adanya pemisahan kode berdasarkan pola MVC, aplikasi menjadi lebih mudah dikembangkan, diuji, serta diperluas jika nantinya ingin menambahkan fitur lain.

DAFTAR PUSTAKA

CodeIgniter Documentation. (2023). *Model-View-Controller (MVC) Pattern*. Diakses dari https://codeigniter.com/userguide3/overview/mvc.html

PHP Official Documentation. (2023). *PHP Manual*. Diakses dari https://www.php.net/manual/

MySQL Documentation. (2023). *MySQL Reference Manual*. Diakses dari https://dev.mysql.com/doc/

Nugroho, Bunafit. (2019). *Membuat Aplikasi Web dengan PHP dan MySQL untuk Pemula*. Yogyakarta: Andi Offset.