Laporan Praktikum

Mata Kuliah Pemrograman WEB

PERTEMUAN 6. TUGAS 1

PRAKTIKUM QUIZ SESSION

Dosen Pengampu: Wildan Aprizall Arifin, S.Pd., M.Kom.

Disusun Oleh:

Nelita Maharani

2307052

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. PENDAHULUAN

CRUD merupakan konsep fundamental dalam dunia pengembangan aplikasi yang mengacu pada empat operasi dasar dalam pengelolaan data: Create (membuat), Read (membaca), Update (memperbarui), dan Delete (menghapus). Konsep ini menjadi pondasi penting dalam hampir setiap aplikasi yang melibatkan manipulasi data, mulai dari aplikasi sederhana hingga sistem yang kompleks.

Dalam konteks pengembangan aplikasi, operasi Create memungkinkan pengguna untuk menambahkan data baru ke dalam sistem. Misalnya, ketika seseorang mendaftar di sebuah platform, data mereka "dibuat" dan disimpan dalam database. Operasi Read digunakan untuk mengambil dan menampilkan data yang telah tersimpan, seperti ketika aplikasi menampilkan daftar pengguna atau detail produk. Update memungkinkan modifikasi data yang sudah ada, contohnya ketika pengguna mengubah profil mereka atau admin memperbarui harga produk. Sementara itu, Delete digunakan untuk menghapus data yang tidak diperlukan lagi dari sistem.

Implementasi CRUD dapat dilakukan melalui berbagai cara, tergantung pada teknologi yang digunakan. Dalam konteks database SQL, operasi Create dilakukan menggunakan perintah INSERT, Read menggunakan SELECT, Update dengan UPDATE, dan Delete menggunakan DELETE. Pada pengembangan API RESTful, operasi-operasi ini umumnya dipetakan ke metode HTTP: POST untuk Create, GET untuk Read, PUT atau PATCH untuk Update, dan DELETE untuk operasi penghapusan.

Ketika mengimplementasikan CRUD, ada beberapa praktik terbaik yang perlu diperhatikan. Pertama, validasi data sangat penting untuk memastikan integritas data yang disimpan. Kedua, aspek keamanan harus diutamakan, termasuk implementasi autentikasi dan autorisasi untuk mencegah akses tidak sah ke operasi CRUD. Ketiga, memberikan umpan balik yang jelas kepada pengguna setelah setiap operasi CRUD sangat penting untuk pengalaman pengguna yang baik. Terakhir, optimasi performa, terutama untuk operasi yang melibatkan data dalam jumlah besar, tidak boleh diabaikan.

Sebagai contoh implementasi, dalam pengembangan aplikasi web menggunakan Node.js dan MySQL, operasi CRUD dapat direalisasikan menggunakan query database yang sesuai. Untuk operasi Create, aplikasi akan menggunakan query INSERT untuk menambahkan data baru. Operasi Read menggunakan SELECT untuk mengambil data yang diperlukan. Update dilakukan dengan query UPDATE untuk memodifikasi data yang sudah ada, dan Delete menggunakan query DELETE untuk menghapus data.

II. ALAT DAN BAHAN

1. Komputer/Laptop
2. Visual Studio Code
3. XAMPP
4. MySQL
5. Chrome

III. LANGKAH KERJA

1. File app.js (File Utama):

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const stockRoutes = require('./routes/stock');
const path = require('path');

const app = express();

app.set('view engine', 'ejs');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
    secret: 'fishstock2024',
    resave: false,
    saveUninitialized: true
}));

app.use(express.static(path.join(__dirname, 'public')));

// Middleware untuk cek login
app.use((req, res, next) => {
     if (!req.session.user && req.path !== '/stock/login' && req.path !== '/stock/register') {
        return res.redirect('/stock/login');
    } else if (req.session.user && req.path === '/') {
        return res.redirect('/stock/dashboard');
    }
    next();
});

app.use('/stock', stockRoutes);

app.get('/', (req, res) => {
    if (req.session.user) {
        return res.redirect('/stock/dashboard');
    } else {
        return res.redirect('/stock/login');
    }
});
```

```
app.listen(3000, () => {
    console.log('Fish Stock Management System running on port 3000');
});
```

File ini berfungsi sebagai titik masuk utama aplikasi yang menggunakan framework Express.js. Di dalamnya terdapat konfigurasi dasar seperti pengaturan template engine EJS dan middleware untuk parsing data. File ini juga menangani manajemen sesi untuk autentikasi pengguna, dimana setiap request akan dicek status loginnya melalui middleware khusus. Jika pengguna belum login, mereka akan diarahkan ke halaman login. File ini juga mengatur routing dasar dan menjalankan server pada port 3000.

2. File config/db.js (Konfigurasi Database):

```
const mysql = require('mysql');

const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'fish_stock_db'
});

db.connect((err) => {
    if (err) throw err;
    console.log('Database connected...');
});

module.exports = db;
```

Berisi konfigurasi koneksi ke database MySQL untuk aplikasi manajemen stok ikan. File ini mendefinisikan parameter koneksi seperti host, username, password, dan nama database. Sebuah koneksi database dibuat menggunakan modul mysql dan diexport agar bisa digunakan oleh file-file lain dalam aplikasi. Ketika koneksi berhasil dibuat, sebuah pesan konfirmasi akan ditampilkan di console.

3. File routes/stock.js (Router Utama):

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const db = require('../config/db');

// CRUD Routes untuk stok ikan
```

```
router.get('/dashboard', (req, res) => {
    if (req.session.user) {
        const query = "SELECT * FROM fish_stock";
        db.query(query, (err, stocks) => {
            if (err) throw err;
            res.render('dashboard', {
                user: req.session.user,
                stocks: stocks
            });
        });
    } else {
        res.redirect('/stock/login');
    }
});

router.post('/add-stock', (req, res) => {
    const { fish_name, quantity, price, supplier } = req.body;
    const query = "INSERT INTO fish_stock (fish_name, quantity, price, supplier)
VALUES (?, ?, ?, ?)";
    db.query(query, [fish_name, quantity, price, supplier], (err, result) => {
        if (err) throw err;
        res.redirect('/stock/dashboard');
    });
});

router.post('/update-stock/:id', (req, res) => {
    const { fish_name, quantity, price, supplier } = req.body;
    const query = "UPDATE fish_stock SET fish_name = ?, quantity = ?, price =
?, supplier = ? WHERE id = ?";
    db.query(query, [fish_name, quantity, price, supplier, req.params.id], (err,
result) => {
        if (err) throw err;
        res.redirect('/stock/dashboard');
    });
});

router.get('/delete-stock/:id', (req, res) => {
    const query = "DELETE FROM fish_stock WHERE id = ?";
    db.query(query, [req.params.id], (err, result) => {
        if (err) throw err;
        res.redirect('/stock/dashboard');
    });
});
```

```
router.get('/login', (req, res) => {
    res.render('login');
});

router.post('/login', (req, res) => {
    const { username, password } = req.body;

    const query = "SELECT * FROM users WHERE username = ?";
    db.query(query, [username], (err, result) => {
        if (err) throw err;

        if (result.length > 0) {
            const user = result[0];
            if (bcrypt.compareSync(password, user.password)) {
                req.session.user = user;
                res.redirect('/stock/dashboard');
            } else {
                res.send('Password salah');
            }
        } else {
            res.send('User tidak ditemukan');
        }
    });
});

router.get('/logout', (req, res) => {
    req.session.destroy();
    res.redirect('/stock/login');
});

module.exports = router;
```

File ini menangani seluruh logika routing untuk manajemen stok ikan. Di dalamnya terdapat berbagai endpoint untuk operasi CRUD (Create, Read, Update, Delete) pada data stok. Setiap route memiliki fungsi khusus, seperti menampilkan dashboard, menambah stok baru, mengupdate informasi stok yang ada, dan menghapus stok. File ini juga menangani proses autentikasi dengan route login dan logout. Setiap operasi database menggunakan koneksi yang telah dikonfigurasi di db.js.

4. File views/dashboard.ejs (Tampilan Dashboard):

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/styles.css">
    <title>Dashboard Stok Ikan</title>
</head>
<body>
    <div class="dashboard-container">
        <header>
            <h1>Manajemen Stok Ikan</h1>
            <p>Selamat datang, <%= user.username %></p>
            <a href="/stock/logout" class="logout-btn">Logout</a>
        </header>

        <div class="stock-form">
            <h2>Tambah Stok Baru</h2>
            <form action="/stock/add-stock" method="POST">
                  <input type="text" name="fish_name" placeholder="Nama Ikan"
required>
                <input type="number" name="quantity" placeholder="Jumlah (kg)"
required>
                 <input type="number" name="price" placeholder="Harga per kg"
required>
                    <input type="text" name="supplier" placeholder="Supplier"
required>
                <button type="submit">Tambah Stok</button>
            </form>
        </div>

        <div class="stock-list">
            <h2>Daftar Stok</h2>
            <table>
                <thead>
                    <tr>
                        <th>Nama Ikan</th>
                        <th>Jumlah (kg)</th>
                        <th>Harga/kg</th>
                        <th>Supplier</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
                    <% stocks.forEach(stock => { %>
                    <tr>
                        <td><%= stock.fish_name %></td>
```

```
                        <td><%= stock.quantity %></td>
                        <td>Rp <%= stock.price %></td>
                        <td><%= stock.supplier %></td>
                        <td>
                                                <button  onclick="showEditForm(<%=
JSON.stringify(stock) %>)">Edit</button>
                            <a href="/stock/delete-stock/<%= stock.id %>"
                                    onclick="return confirm('Yakin hapus stok
ini?')">Hapus</a>
                        </td>
                    </tr>
                    <% }); %>
                </tbody>
            </table>
        </div>
    </div>

    <div id="editModal" class="modal">
        <div class="modal-content">
            <h2>Edit Stok</h2>
            <form id="editForm" action="/stock/update-stock/" method="POST">
                <input type="text" name="fish_name" id="edit_fish_name">
                <input type="number" name="quantity" id="edit_quantity">
                <input type="number" name="price" id="edit_price">
                <input type="text" name="supplier" id="edit_supplier">
                <button type="submit">Update</button>
                <button type="button" onclick="closeEditModal()">Batal</button>
            </form>
        </div>
    </div>
</body>
</html>
```

Merupakan template EJS untuk halaman utama aplikasi yang menampilkan interface manajemen stok ikan. Halaman ini memiliki beberapa komponen utama seperti header yang menampilkan informasi pengguna, form untuk menambah stok baru, dan tabel yang menampilkan daftar stok yang ada. Setiap baris tabel dilengkapi dengan tombol untuk edit dan hapus data. Template ini menggunakan sintaks EJS untuk menampilkan data dinamis dari server.

5. File views/login.ejs (Halaman Login):

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

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/styles.css">
    <title>Login - Manajemen Stok Ikan</title>
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        <form action="/stock/login" method="POST">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Password</label>
            <input type="password" id="password" name="password" required>

            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

Template untuk halaman login yang menampilkan form autentikasi sederhana. Halaman ini memiliki field untuk username dan password, serta tombol submit untuk proses login. Desainnya dibuat minimalis namun fungsional, dengan tampilan yang responsif. Form ini akan mengirimkan data ke endpoint login untuk diproses.

6. File public/styles.css (Stylesheet):

```css
body {
    font-family: Arial, sans-serif;
    background-color: #e6f3ff;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
```

```css
}

.dashboard-container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: #004d99;
    color: white;
    padding: 20px;
    border-radius: 8px;
    margin-bottom: 20px;
}

.stock-form {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    margin-bottom: 20px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.stock-list {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #ddd;
```

```
}

th {
    background-color: #f2f2f2;
}

button {
    background-color: #004d99;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #003366;
}

.logout-btn {
    background-color: #ff4444;
    color: white;
    padding: 8px 16px;
    border-radius: 4px;
    text-decoration: none;
}

.logout-btn:hover {
    background-color: #cc0000;
}

input {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
}

.modal {
    display: none;
    position: fixed;
    top: 0;
```

```
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0,0,0,0.5);
}

.modal-content {
    background-color: white;
    margin: 15% auto;
    padding: 20px;
    width: 80%;
    max-width: 500px;
    border-radius: 8px;
}
```

File CSS ini mengatur seluruh tampilan visual aplikasi. Di dalamnya terdapat pengaturan untuk layout, warna, spacing, dan responsivitas. Stylesheet ini menggunakan pendekatan mobile-first dengan media queries untuk tampilan yang responsif. Beberapa komponen yang diatur termasuk container utama, tabel data, form input, tombol, dan modal dialog. Warna-warna yang digunakan disesuaikan dengan tema maritim, menggunakan berbagai gradasi biru.

7. Struktur Database:

```
-- Active: 1727361777082@@127.0.0.1@3306
CREATE DATABASE fish_stock_db
    DEFAULT CHARACTER SET = 'utf8mb4';
-- Active: 1727361777082@@127.0.0.1@3306@fish_stock_db
CREATE TABLE fish_stock (
    id INT AUTO_INCREMENT PRIMARY KEY,
    fish_name VARCHAR(100) NOT NULL,
    quantity FLOAT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    supplier VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Database menggunakan dua tabel utama: fish_stock untuk menyimpan informasi stok ikan dan users untuk data pengguna. Tabel fish_stock menyimpan detail seperti nama ikan, jumlah, harga, dan supplier,

sementara tabel users menyimpan kredensial login pengguna dengan password yang dienkripsi menggunakan bcrypt. Setiap tabel dilengkapi dengan timestamp untuk tracking waktu pembuatan data.

8. Alur Kerja Aplikasi:

Ketika aplikasi dijalankan, server Express akan start dan mendengarkan request pada port 3000. Pengguna harus login terlebih dahulu sebelum dapat mengakses dashboard. Setelah login berhasil, pengguna dapat melakukan berbagai operasi manajemen stok seperti menambah, melihat, mengubah, dan menghapus data stok ikan. Setiap operasi akan langsung memperbarui tampilan dan database.

IV. KESIMPULAN

Hasil dari materi praktikum ini adalah bahwa kita telah membangun dan mengelola aplikasi web sederhana menggunakan Node.js sebagai framework, Express sebagai basis data, dan bcrypt.js untuk hashing kata sandi untuk keamanan. Aplikasi ini memiliki fitur autentikasi pengguna dasar, seperti registrasi, login, logout, dan tampilan profil pengguna. Untuk menjalankannya, kami membuat jalur untuk menjalankan berbagai fungsi tersebut, menggunakan express-session untuk mengelola sesi pengguna, dan menerapkan middleware untuk memastikan hanya pengguna yang telah login yang dapat mengakses halaman tertentu. Halaman frontend dibuat menggunakan template engine EJS dan diformat dengan CSS untuk tampilan yang bersih dan responsif. Praktikum ini mengajarkan cara membuat aplikasi autentikasi yang aman dan berfungsi dengan menggabungkan berbagai komponen.

Link Github:

https://github.com/maharaninelita/praktikSession/upload/main/PraktikSession_pemweb