

# Muhammad Reza

## NPM. 2221101826

### BAB VI:

## Bekerja dengan Multi-Form

### 1. Tujuan Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

1. Menjelaskan konsep dasar multi-form: Memahami struktur aplikasi dengan lebih dari satu form dan kegunaannya dalam memisahkan fungsi seperti login, registrasi, dan dashboard
2. Membuat dan mengelola beberapa form: Membuat LoginForm, RegisterForm, dan DashboardForm secara terpisah serta mengatur inisialisasi dan tampilannyaMenavigasi antar form menggunakan metode yang tepat.
3. Menavigasi antar form: Menggunakan metode Hide() dan Show() untuk berpindah antar form tanpa menghancurkan objek, serta memilih pola navigasi yang sesuai
4. Mengelola komunikasi antar form: Melakukan passing data melalui constructor, menggunakan kembali instance form yang sudah ada, dan mengelola siklus hidup form
5. Mengimplementasikan best practices: Menerapkan separation of concerns, validasi input, penanganan error, keamanan password, dan manajemen memori yang efisien

### 2. Pengantar Teori

#### 2.1 Kegunaan Multi-Form dalam Aplikasi Desktop

Dalam pengembangan aplikasi desktop modern, konsep single form application (aplikasi dengan satu form saja) sangat terbatas dan tidak praktis untuk aplikasi yang kompleks. Bayangkan jika semua fitur aplikasi seperti login, registrasi, dashboard, pengaturan, dan laporan harus ditampung dalam satu form yang sama - hasilnya akan sangat membingungkan dan sulit digunakan.

Multi-form application adalah solusi yang memungkinkan aplikasi memiliki beberapa jendela (form) yang masing-masing memiliki fungsi dan tanggung jawab yang spesifik.

Keuntungan Multi-Form:

1. Modularitas yang Tinggi
  - a. Setiap form fokus pada satu tanggung jawab spesifik (Single Responsibility Principle)
  - b. Memudahkan maintenance dan debugging
  - c. Kode lebih terorganisir dan mudah dipahami

2. User Experience yang Lebih Baik
  - a. Navigasi yang intuitif dan logical flow
  - b. Interface yang tidak overloaded dengan terlalu banyak kontrol
  - c. User dapat fokus pada satu task dalam satu waktu
3. Skalabilitas dan Fleksibilitas
  - a. Penambahan fitur baru cukup dengan membuat form baru
  - b. Perubahan pada satu form tidak mempengaruhi form lain
  - c. Memungkinkan pengembangan paralel oleh tim
4. Reusability
  - a. Form dapat digunakan kembali dalam konteks yang berbeda
  - b. Komponen dapat dibagi antar form

## 2.2 Konsep Dasar Multi-Form dalam Windows Forms

Dalam Windows Forms, setiap form adalah turunan dari kelas `System.Windows.Forms.Form`. Ini berarti setiap form mewarisi semua properti dan method dasar yang diperlukan untuk mengelola jendela aplikasi.

```
// Setiap form adalah turunan dari Form class
public partial class LoginForm : Form
{
    // LoginForm memiliki semua kemampuan dasar Form
    // Fungsionalitas khusus untuk login
}
public partial class RegisterForm : Form
{
    // RegisterForm memiliki fungsionalitas khusus untuk registrasi
}
public partial class DashboardForm : Form
{
    // DashboardForm memiliki fungsionalitas khusus untuk dashboard
}
```

Setiap form dalam aplikasi Windows Forms merupakan kelas yang mewarisi dari `System.Windows.Forms.Form`. Dengan pewarisan ini, setiap form secara otomatis memiliki fungsi dasar seperti tampilan jendela, kontrol ukuran, dan manajemen event. Form seperti `LoginForm`, `RegisterForm`, dan `DashboardForm` dibuat sebagai kelas terpisah yang masing-masing memiliki tugas spesifik, memungkinkan aplikasi lebih terstruktur, mudah dikelola, dan sesuai prinsip pemrograman berbasis objek.

## 3. Contoh Kode Program

Dalam proyek `LoginFormApp`, konsep multi-form diimplementasikan secara nyata:

- `LoginForm` sebagai form utama untuk autentikasi.
- `RegisterForm` sebagai form untuk pendaftaran akun baru.
- `DashboardForm` sebagai form tujuan setelah login berhasil.

Navigasi antar form diatur menggunakan method seperti `.Show()` dan `.Hide()`, serta penggunaan konstruktor untuk passing data antar form (`DashboardForm(username)`).

Berikut adalah langkah-langkah detail dan penjelasan kode yang membangun fungsionalitas multi-form pada aplikasi login ini.

## Langkah 1: Membuat Entry Point Aplikasi (Program.cs)

Setiap aplikasi Windows Forms dimulai dari satu titik. File ini menentukan form mana yang pertama kali muncul.

```
// File Program.cs
static void Main()
{
    Application.EnableVisualStyles();
    Application.Run(new LoginForm());
}
```

Penjelasan:

- Application.Run() memulai aplikasi dan menampilkan form yang ditentukan
- LoginForm dipilih sebagai form pertama karena user harus login terlebih dahulu
- Ini adalah gerbang masuk ke seluruh sistem aplikasi
- Method ini akan terus berjalan sampai aplikasi ditutup

## Langkah 2: Membuat Form Login Dasar

Form login adalah interface pertama yang dilihat user. Kita perlu membuat tampilan dan fungsi dasar.

```
// File LoginForm.cs
public partial class LoginForm : Form
{
    private TextBox txtUsername, txtPassword;
    private Button btnLogin, btnRegister;

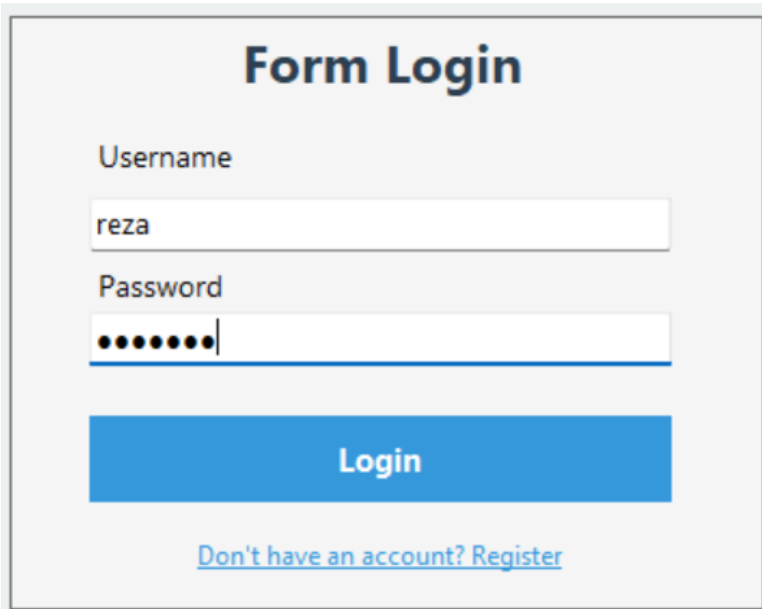
    public LoginForm()
    {
        InitializeComponent();
        SetupForm();
    }

    private void SetupForm()
    {
        this.Text = "Login";
        this.Size = new Size(400, 300);
        this.StartPosition = FormStartPosition.CenterScreen;
    }
}
```

Penjelasan:

- Partial Class: Memungkinkan pemisahan kode designer dan logic
- Control Declaration: Mendeklarasikan kontrol yang akan digunakan (TextBox, Button)
- Constructor: Tempat inisialisasi form dan pengaturan awal
- SetupForm(): Method terpisah untuk konfigurasi tampilan form
- CenterScreen: Membuat form muncul di tengah layar untuk UX yang baik

## Tampilan Form Login



Pada langkah ini, tata letak form login harus sudah dibuat melalui Designer:

- Dua TextBox untuk username dan password
- Dua Button: Login dan Register
- Label sebagai keterangan input

## Langkah 3: Implementasi Navigasi ke Form Register

User yang belum punya akun perlu bisa pindah ke form registrasi.

```
// File LoginForm.cs
private void btnRegister_Click(object sender, EventArgs e)
{
    this.Hide();
    RegisterForm registerForm = new RegisterForm();
    registerForm.Show();
}
```

Penjelasan:

- Method btnRegister\_Click ditambahkan di dalam kelas LoginForm.
- Hide() vs Close(): Hide menyembunyikan form tapi tetap di memory, sehingga bisa ditampilkan lagi
- New Instance: Membuat objek baru dari RegisterForm
- Show(): Menampilkan form register ke user
- Event Handler: Method ini dipanggil saat tombol register diklik
- Form login tetap ada di background, siap ditampilkan kembali

#### Langkah 4: Membuat Form Register dengan Validasi

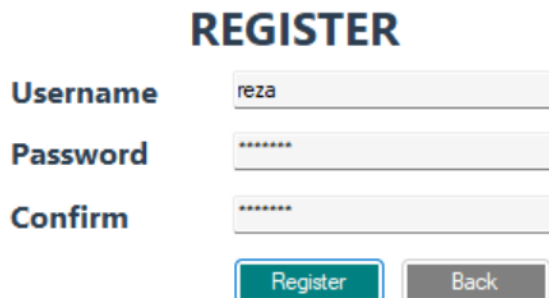
Form register memungkinkan user membuat akun baru dengan validasi yang proper.

```
// File RegisterForm.cs
public partial class RegisterForm : Form
{
    private TextBox txtUsername, txtPassword, txtConfirmPassword;
    private Button btnRegister, btnBack;
    private void btnRegister_Click(object sender, EventArgs e)
    { // Validasi input
        if (string.IsNullOrEmpty(txtUsername.Text) ||
            string.IsNullOrEmpty(txtPassword.Text))
        {
            MessageBox.Show("Semua field harus diisi!");
            return; }
        if (txtPassword.Text != txtConfirmPassword.Text)
        {
            MessageBox.Show("Password tidak cocok!");
            return; }
        // Simpan ke database
        if (SaveUserToDatabase())
        {
            MessageBox.Show("Registrasi berhasil!");
            BackToLogin();
        }
    }
    private void BackToLogin()
    {
        this.Hide();
        new LoginForm().Show();
    }
}
```

Penjelasan:

- Input Validation: Memastikan semua field terisi dan password cocok
- Early Return: Menggunakan return untuk menghentikan eksekusi jika validasi gagal
- User Feedback: MessageBox memberikan informasi yang jelas kepada user
- Separation of Concerns: Method BackToLogin() terpisah untuk navigasi
- Database Integration: Method SaveUserToDatabase() menangani penyimpanan data
- Setelah registrasi berhasil, user diarahkan kembali ke form login

#### Tampilan Form Register



The image shows a web form titled "REGISTER" in a large, bold, blue font. Below the title, there are three labels in blue: "Username", "Password", and "Confirm". Each label is followed by a text input field. The "Username" field contains the text "reza". The "Password" and "Confirm" fields contain masked characters (dots). At the bottom of the form, there are two buttons: a green "Register" button and a grey "Back" button.

Pada langkah ini, tata letak form register harus sudah dibuat melalui Designer:

- Tiga TextBox: username, password, konfirmasi password
- Dua Button: Register dan Kembali
- Label sebagai keterangan input

### Langkah 5: Membuat Database

Sebelum menghubungkan aplikasi ke database, struktur database harus dibuat terlebih dahulu, dapat dimulai dengan masuk ke Command Prompt kemudian masuk ke direktori xampp untuk mengakses mysql

```
//Query SQL
-- Membuat database
CREATE DATABASE loginapp;

-- Menggunakan database tersebut
USE loginapp;

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

-- Data uji coba (opsional)
INSERT INTO users (username, password) VALUES ('admin', 'admin123');
```

Penjelasan:

- Nama database disesuaikan dengan connectionString di kode (loginapp)
- Kolom password bertipe VARCHAR(255) untuk menampung hasil hash
- AUTO\_INCREMENT dan PRIMARY KEY memastikan setiap user memiliki ID unik
- Data uji coba membantu dalam proses testing tanpa harus registrasi manual

### Langkah 6: Implementasi Koneksi Database

Database diperlukan untuk menyimpan dan memverifikasi data user.

```
//File Database Helper.cs
public class DatabaseHelper
{
    private string connectionString = "Server=localhost;Database=loginapp;Uid=root;Pwd=";
    public bool SaveUser(string username, string password)
    {
        try
        {
            using (MySqlConnection conn = new MySqlConnection(connectionString))
            {
                conn.Open();
                string hashedPassword = HashPassword(password);
                string query = "INSERT INTO users (username, password) VALUES (@username, @password)";
                MySqlCommand cmd = new MySqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@username", username);
            }
        }
    }
}
```

```

        cmd.Parameters.AddWithValue("@password", hashedPassword);
        cmd.ExecuteNonQuery();
        return true;
    }
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message);
    return false;
}
}
private string HashPassword(string password)
{
    using (SHA256 sha256 = SHA256.Create())
    {
        byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
        return Convert.ToBase64String(bytes);
    }
}
}
}

```

Penjelasan:

- Separate Class: DatabaseHelper terpisah untuk reusability dan maintainability
- Connection String: Berisi informasi koneksi ke database MySQL
- Using Statement: Memastikan koneksi database ditutup otomatis
- Parameterized Query: Mencegah SQL injection dengan menggunakan parameter
- Password Hashing: SHA256 untuk keamanan, password tidak disimpan dalam bentuk plain text
- Error Handling: Try-catch untuk menangani kemungkinan error database
- Return Boolean: Memberikan feedback apakah operasi berhasil atau gagal

## Langkah 7: Implementasi Proses Login dengan Verifikasi

Setelah user register, mereka perlu bisa login dengan kredensial yang telah dibuat.

```

//File LoginForm.cs
private void btnLogin_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text;
    string password = txtPassword.Text;
    if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
    {
        MessageBox.Show("Username dan password harus diisi!");
        return;
    }
    DatabaseHelper db = new DatabaseHelper();
    if (db.VerifyUser(username, password))
    {
        // Login berhasil - pindah ke dashboard
        this.Hide();
        DashboardForm dashboard = new DashboardForm(username);
        dashboard.Show();
    } else {
        MessageBox.Show("Username atau password salah!");
    }
}
}

```

Penjelasan:

- Input Validation: Memastikan field tidak kosong sebelum proses verifikasi
- Database Verification: Menggunakan DatabaseHelper untuk memverifikasi kredensial
- Conditional Navigation: Hanya pindah ke dashboard jika login berhasil
- Data Passing: Mengirim username ke DashboardForm melalui constructor
- User Feedback: Pesan error yang jelas jika login gagal
- Form login disembunyikan setelah login berhasil

## Langkah 8: Menambahkan Method Verifikasi Database

Method untuk memverifikasi kredensial user saat login.

```
//File DatabaseHelper.cs
public bool VerifyUser(string username, string password)
{
    try
    {
        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            conn.Open();
            string hashedPassword = HashPassword(password);
            string query = "SELECT COUNT(*) FROM users WHERE username = @username AND password = @password";
            MySqlCommand cmd = new MySqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@username", username);
            cmd.Parameters.AddWithValue("@password", hashedPassword);
            int count = Convert.ToInt32(cmd.ExecuteScalar());
            return count > 0;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error verifikasi: " + ex.Message);
        return false;
    }
}
```

Penjelasan:

- ExecuteScalar(): Mengembalikan nilai tunggal (COUNT) dari query
- Password Matching: Hash password input dan bandingkan dengan yang tersimpan di database
- Boolean Return: True jika user ditemukan, False jika tidak
- Security: Menggunakan parameterized query dan hashed password
- Error Handling: Mengembalikan false jika terjadi error database



## Langkah 9: Membuat Dashboard dengan Data Personal

Dashboard adalah form utama setelah user berhasil login, menampilkan informasi personal.

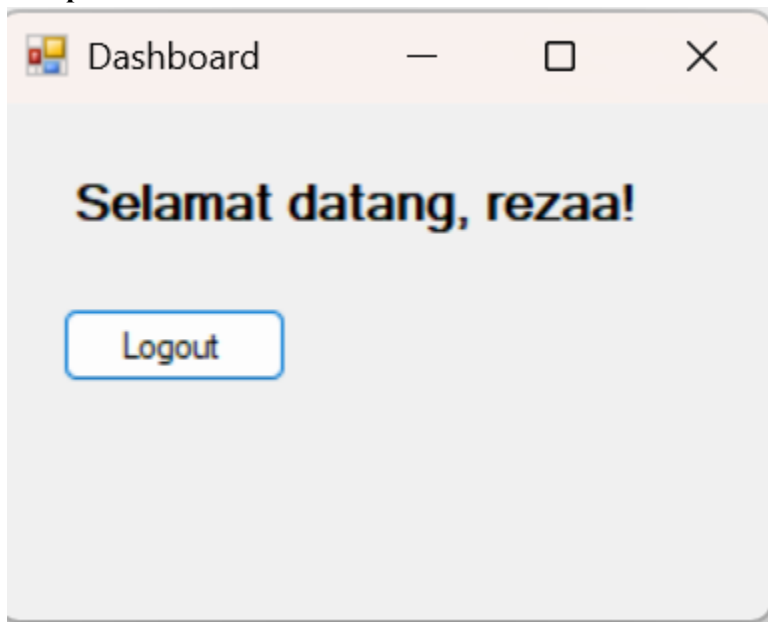
```
//File DashboardForm.cs
public partial class DashboardForm : Form
{
    private string currentUser;
    private Label lblWelcome;
    private Button btnLogout;
    public DashboardForm(string username)
    {
        InitializeComponent();
        currentUser = username;
        SetupDashboard();
    }
    private void SetupDashboard()
    {
        lblWelcome.Text = $"Selamat datang, {currentUser}!";
        this.Text = "Dashboard - " + currentUser;
    }
    private void btnLogout_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Yakin ingin logout?", "Konfirmasi",
                                                MessageBoxButtons.YesNo);

        if (result == DialogResult.Yes)
        {
            this.Close();
            // Tampilkan kembali form login
            foreach (Form form in Application.OpenForms)
            {
                if (form is LoginForm)
                {
                    form.Show();
                    return;
                }
            }
            // Jika LoginForm tidak ditemukan, buat yang baru
            new LoginForm().Show();
        }
    }
}
```

Penjelasan:

- Constructor Parameter: Menerima username dari form login untuk personalisasi
- Data Storage: Menyimpan username dalam variabel instance untuk digunakan di seluruh form
- UI Personalization: Menampilkan nama user di label welcome dan title bar
- Confirmation Dialog: Meminta konfirmasi sebelum logout untuk mencegah logout tidak sengaja
- Smart Navigation: Mencari LoginForm yang sudah ada di memory sebelum membuat yang baru
- Form Recovery: Menampilkan kembali form login yang sebelumnya di-hide
- Fallback Mechanism: Membuat LoginForm baru jika yang lama tidak ditemukan

### Tampilan Dashboard



Username yang didaftarkan pada register dan melakukan login sehingga terkoneksi dengan database, berhasil dipanggil dan ditampilkan pada halaman dashboard.

### Kesimpulan

Modul ini membahas penerapan konsep multi-form dalam aplikasi desktop menggunakan C# Windows Forms secara sistematis dan berurutan. Dimulai dari pembuatan entry point di Program.cs, pembentukan tampilan dan logika pada LoginForm, RegisterForm, hingga DashboardForm, serta implementasi navigasi antar form dengan metode Hide() dan Show(). Penekanan diberikan pada prinsip modularitas, pemisahan logika dan desain melalui partial class, serta manajemen siklus hidup form untuk menciptakan aplikasi yang terstruktur dan mudah dikelola.

Integrasi database menjadi bagian penting dalam modul ini, dengan pembuatan tabel users dan implementasi koneksi menggunakan MySqlConnection. Proses registrasi dan login dilengkapi validasi input, hashing password menggunakan SHA256, dan parameterized query untuk keamanan. Data pengguna yang berhasil login dikirim ke DashboardForm melalui constructor, menunjukkan teknik passing data yang aman dan efisien, sehingga menghasilkan aplikasi yang tidak hanya fungsional, tetapi juga aman dan user-friendly.