



IT DEL

Kelompok-15

SISTEM PENGELOLAAN EVENT ORGANIZER

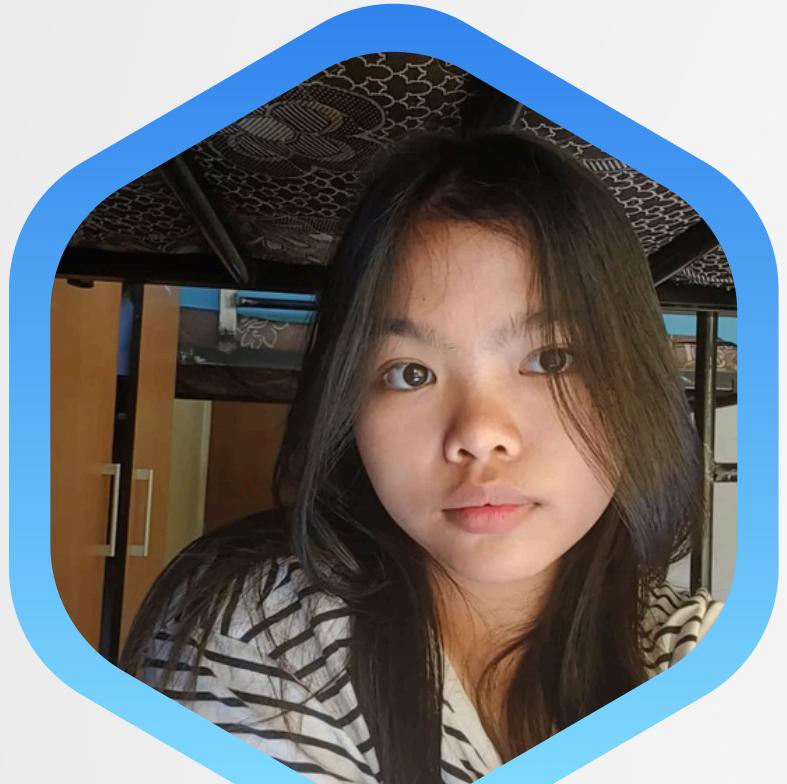
PRESENTATION PBO 2024

Presented For

Rudy Chandra, S.Kom., M.Kom



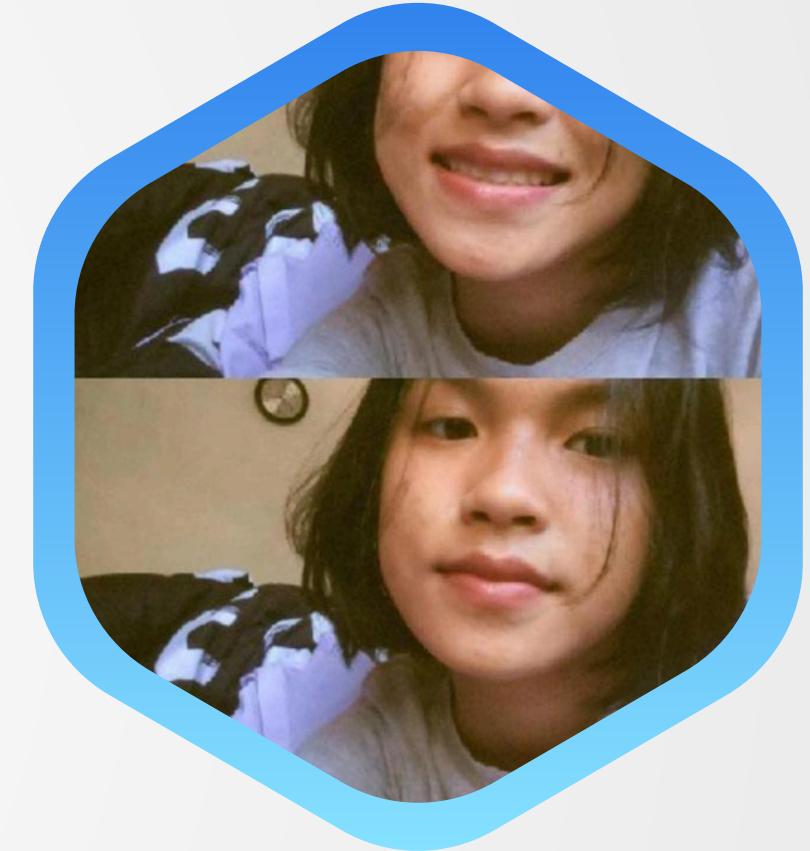
Meet Our Team



Marshanda Kasih Simangunsong
11323020



Frangklyn Aldo Ignatia Lumbantoruan
11323010



Febyanti Hutahaean
11323055

Sistem Pengelolaan Event Organizer



Deskripsi

Sistem Pengelolaan Event Organizer (EO) adalah sebuah sistem yang dirancang untuk mempermudah pengelolaan data terkait klien, acara, dan jadwal acara secara terpusat. Sistem ini akan dilengkapi dengan antarmuka yang user-friendly dan fitur yang komprehensif untuk mendukung kebutuhan operasional EO.

Tujuan

1. Memudahkan Event Organizer dalam menyimpan dan mengakses informasi klien serta acara tanpa kendala.
2. Memungkinkan Admin untuk melihat jumlah acara yang belum diselesaikan.
3. Mengurangi potensi kesalahan dalam pengelolaan data yang dilakukan secara manual.
4. Meningkatkan produktivitas tim Event Organizer.

Bisnis Proses Sistem

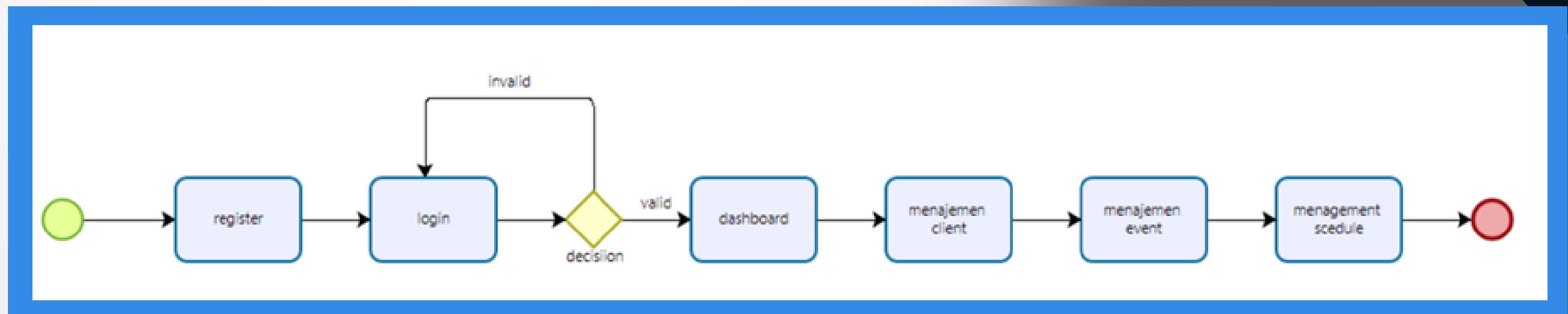
Penjelasan

Dalam sistem ini, hanya terdapat satu peran, yaitu Admin, yang merupakan pegawai dari Event Organizer. Admin memiliki tanggung jawab utama untuk mencatat data terkait klien, event, dan jadwal acara. Sebelum melakukan pengolahan data, Admin diwajibkan untuk melakukan proses login terlebih dahulu. Langkah ini bertujuan untuk menjaga keamanan sistem, memastikan bahwa hanya pengguna yang memiliki otorisasi yang dapat mengakses dan mengelola data.



Fitur Utama

1. Register
2. Login
3. Mengelola Client
4. Mengelola Event
5. Mengelola Schedule



Penerapan Konsep OOP



1. Abstraction

Abstraksi mengacu pada menyembunyikan detail implementasi dan hanya menampilkan fungsionalitas yang diperlukan. Pada proyek ini, kelas ClientController berfungsi sebagai pengendali antara tampilan dan model data, menyembunyikan proses bisnis dan komunikasi dengan database. Misalnya, saat menambahkan client, pengguna tidak perlu tahu bagaimana data disimpan dalam database, hanya perlu mengisi form dan klik tombol simpan.

Contoh kode terkait abstraksi: [Lihat gambar disamping](#)

```
private void updateClient() {  
    if (selectedClient == null) return;  
  
    String nama = nameField.getText().trim();  
    String kontak = contactField.getText().trim();  
    String alamat = addressField.getText().trim();  
  
    if (nama.isEmpty() || kontak.isEmpty() || alamat.isEmpty()) {  
        showAlert(title: "Validasi", message: "Harap isi semua field!");  
        return;  
    }  
}
```

2. Encapsulation (Enkapsulasi)

```
public class Client {  
    private IntegerProperty number;   
    private IntegerProperty id;   
    private StringProperty name;   
    private StringProperty contact;   
    private StringProperty address;  
  
    public Client(String name, String contact, String address) {  
        this.number = new SimpleIntegerProperty();  
        this.id = new SimpleIntegerProperty();  
        this.name = new SimpleStringProperty(string:name);  
        this.contact = new SimpleStringProperty(string:contact);  
        this.address = new SimpleStringProperty(string:address);  
    }   
    // Getter dan Setter biasa  
    public int getNumber() {  
        return number.get();  
    }  
  
    public void setNumber(int number) {  
        this.number.set(value: number);  
    }  
}
```

Enkapsulasi adalah teknik untuk menyembunyikan data (fields) dalam kelas dan hanya memberikan akses melalui metode getter dan setter. Pada kelas Client, data pribadi seperti nama, kontak, dan alamat disembunyikan di dalam objek Client. Akses dilakukan melalui method getName(), setName(), dan lainnya.

Contoh kode terkait enkapsulasi:
Lihat Gambar Disamping

3. Inheritance (Pewarisan)

Inheritance atau pewarisan dalam OOP memungkinkan sebuah kelas untuk mewarisi sifat dan metode dari kelas lainnya. Dalam proyek ini, kelas ScheduleController mewarisi sifat dan metode dari kelas abstrak BaseController. Dengan menggunakan inheritance, metode seperti navigateToView dan showAlert dapat digunakan kembali oleh ScheduleController tanpa perlu didefinisikan ulang. Hal ini membantu mengurangi duplikasi kode dan mempermudah pengelolaan fitur yang bersifat umum.

Contoh kode terkait inheritance:

```
interface Controller extends Initializable {  
    void setupButtons();  
    void loadData();  
}  
  
public class ScheduleController extends BaseController implements Controller {  
    @FXML  
}
```

4. Polymorphism (Polimorfisme)

```
// ·Tombol ·Simpan¶  
·btnSaveClient.setOnAction (event ·-> ·saveClient () ) ;¶  
  
// ·Tombol ·Update¶  
·btnUpdateClient.setOnAction (event ·-> ·updateClient () ) ;¶  
  
// ·Tombol ·Delete¶  
·btnDeleteClient.setOnAction (event ·-> ·deleteClient () ) ;¶  
  
// ·Tombol ·Batal¶  
·btnCancelClient.setOnAction (event ·-> ·{ ¶  
·····addClientPopup.setVisible (value: false) ;¶  
·····clearFields () ;¶  
} ) ;¶
```

Polimorfisme memungkinkan objek dari kelas yang berbeda untuk diakses melalui antarmuka yang sama. Pada contoh proyek ini, polimorfisme diterapkan pada tombol-tombol yang memiliki tindakan berbeda meskipun menggunakan event handler yang serupa. Misalnya, tombol simpan, update, dan delete memiliki event handler yang berbeda namun mereka berbagi struktur kode yang mirip.

Contoh kode terkait polimorfisme:

Lihat Gambar disamping

Semua tombol menggunakan event handler dengan pola yang mirip meskipun fungsionalitasnya berbeda.

5. Interface (Antarmuka)

Interface dalam OOP mendefinisikan kontrak atau perilaku yang harus diimplementasikan oleh suatu kelas. Dalam proyek ini, ScheduleController mengimplementasikan interface Controller, yang mendefinisikan metode setupButtons dan loadData. Dengan cara ini, setiap kelas yang mengimplementasikan interface Controller harus menyediakan implementasi spesifik untuk kedua metode tersebut, memastikan konsistensi antar kelas yang berbeda. Contoh kode terkait interface: **Lihat gambar disamping**

```
interface ·Controller ·extends ·Initializable {  
    ··· void ·setupButtons () ;  
    ··· void ·loadData () ;  
}
```

6. Exception Handling (Penanganan Eksepsi)

Penanganan eksepsi digunakan untuk menangani kesalahan yang dapat terjadi selama eksekusi program, seperti kesalahan koneksi database atau kesalahan dalam query. Pada kode ini, kami menangani beberapa eksepsi seperti SQLException saat berinteraksi dengan database.

Contoh kode terkait penanganan eksepsi:
Lihat gambar disamping.

Jika terjadi kesalahan dalam query, eksepsi akan ditangani dan ditampilkan ke pengguna melalui showAlert().

```
try {¶
    String query = "INSERT INTO clients (name, contact, address) VALUES (?, ?, ?);¶
    PreparedStatement statement = connection.prepareStatement(string:query, i: Statement.RETURN_GENERATED_KEYS);¶
    statement.setString(i: 1, string:nama);¶
    statement.setString(i: 2, string:kontak);¶
    statement.setString(i: 3, string:alamat);¶

    int rowsInserted = statement.executeUpdate();¶
    if (rowsInserted > 0) {¶
        ResultSet generatedKeys = statement.getGeneratedKeys();¶
        if (generatedKeys.next()) {¶
            int id = generatedKeys.getInt(i: 1);¶
            Client newClient = new Client(name: nama, contact: kontak, address: alamat);¶
            newClient.setNumber(number:id);¶
            clientData.add(e: newClient);¶
        }
        clearFields();¶
        addClientPopup.setVisible(value: false);¶
    }
} catch (SQLException e) {¶
    showAlert(title: "Error Menyimpan", message: e.getMessage());¶
}
```

Method dalam class



1 Method pada class loginController

method	deskripsi	parameter
handleLogin()	Mengambil data dari form login, memvalidasi input, dan memeriksa ke basis data untuk proses autentikasi.	-
goToMainView()	Berpindah ke halaman utama MainView.fxml jika login berhasil.	-
goToRegister()	Berpindah ke halaman registrasi Register.fxml jika pengguna ingin mendaftar.	-
showTemporaryNotification(String message, String type, double durationInSeconds, Runnable onComplete)	Menampilkan notifikasi sementara dengan pesan tertentu, tipe notifikasi (sukses/gagal), dan durasi tampilan.	message, type, durationInSeconds, onComplete

Pada kelas loginController, terdapat empat method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.

2 Method pada class registerController

Nama Method	parameter	deskripsi
handleRegister()	-	Memproses registrasi pengguna dengan memvalidasi input, memeriksa username, dan menyimpan ke database.
isUsernameExists(String username)	username: String	Mengecek apakah username sudah ada di database.
goToLogin()	-	Mengarahkan pengguna ke halaman Login.
showAlert()	type: AlertType, title: String, content: String	Menampilkan pesan kesalahan atau pemberitahuan ke pengguna.
showTemporaryNotification()	message: String, durationInSeconds: double, onComplete: Runnable	Menampilkan notifikasi sementara dengan durasi tertentu dan tindakan setelah selesai.

Pada kelas registerController, terdapat 5 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.

3 Method pada class clientController

Nama Method	Parameter	
<u>initialize()</u>	-	Mengatur kolom TableView , tombol dan data awal.
<u>btnAddClientTop.setOnAction()</u>	Event action dari tombol	Membuka form tambah client saat tombol ditekan.
<u>btnSaveClient.setOnAction()</u>	Event action dari tombol	Menyimpan data client dari form ke ObservableList
<u>btnCancelClient.setOnAction()</u>	Event action dari tombol	Membatalkan form tambah client tanpa menyimpan data.

Pada kelas clientController, terdapat 4 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.

4 Method pada class eventController



Nama Method	Parameter	Deskripsi
<u>initialize()</u>	-	Method ini dipanggil saat tampilan UI di-load. Mengatur kolom tabel agar sesuai dengan properti di kelas Event dan memuat data dari database.
<u>loadDataFromDatabase()</u>	-	Mengambil data event dari database MySQL menggunakan koneksi JDBC. Data event dimasukkan ke dalam ObservableList dan ditampilkan di tabel UI.
<u>setCellValueFactory()</u>	Properti dari TableColumn	Menantikan kolom-kolom pada UI dengan atribut dari kelas Event, sehingga data dari database dapat ditampilkan.

Pada kelas eventController, terdapat 3 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.



5 Method pada class scheduleController



Nama Method	Parameter	DEskripsi
<u>initialize()</u>	-	Method ini dipanggil saat tampilan UI di-load. Digunakan untuk inisialisasi awal seperti pengaturan kolom tabel atau pengaturan UI lainnya.
<u>showAddSchedulePopup()</u>	-	Method ini membuat popup form untuk menambahkan jadwal menjadi terlihat (setVisible(true)).
<u>hideAddSchedulePopup()</u>	-	Method ini menyembunyikan popup form dengan mengatur visibilitas menjadi false (setVisible(false)).

Pada kelas schedule Controller, terdapat 3 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.



6 Method pada class mainController

Nama Method	Parameter	Deskripsi
<u>initialize()</u>	-	Method yang dipanggil saat aplikasi dijalankan untuk menginisialisasi komponen UI dan memuat data dari database.
<u>connectToDatabase()</u>	-	Membuka koneksi ke database menggunakan utility yang sudah disiapkan.
<u>loadTotalCounts()</u>	-	Memuat data jumlah total pengguna, klien, dan event dari database dan menampilkannya di label yang sesuai.
<u>loadEventStatusChart()</u>	-	Memuat data status event dari database dan menampilkan data tersebut dalam bentuk grafik PieChart .
<u>loadTableView()</u>	-	Memuat daftar nama klien dari database dan menampilkannya dalam TableView .

table 6 method MainController

Pada kelas mainController, terdapat 5 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.

7 Method pada class client



Nama Method	Parameter	Deskripsi
<code>Client(String name, String contact, String address)</code>	<code>name,</code> <code>contact,</code> <code>address</code>	Constructor untuk membuat objek baru dari class Client.
<code>getName()</code>	-	Mengambil data nama client.
<code>setName(String name)</code>	<code>name</code> (String)	Mengatur data nama client.
<code>getContact()</code>	-	Mengambil data kontak client.
<code>setContact(String contact)</code>	<code>contact</code> (String)	Mengatur data kontak client.
<code>getAddress()</code>	-	Mengambil data alamat client.
<code>setAddress(String address)</code>	<code>address</code> (String)	Mengatur data alamat client.

Pada kelas client, terdapat 7 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.



8 Method pada class event



Nama Method	Parameter	Deskripsi
<u>initialize()</u>	-	Dipanggil setelah file FXML dimuat, digunakan untuk inisialisasi elemen UI.
<u>showAddSchedulePopup()</u>	-	Menampilkan popup untuk menambahkan jadwal baru.
<u>hideAddSchedulePopup()</u>	-	Menyembunyikan popup penambahan jadwal.
<u>loadScheduleData(List<Schedule> scheduleList)</u>	<u>scheduleList</u> (List<Schedule>)	Memuat data jadwal ke dalam TableView.

Pada kelas event, terdapat 4 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.



9 Method pada class schedule

Nama Method	Parameter	Deskripsi
getHour()	-	Mengambil nilai jam.
setHour(int hour) ()	hour (int)	Menetapkan nilai jam
getDay1() ()	-	Mengambil nilai jadwal untuk hari 1.
setDay1(String day1)	day1 (String)	Menetapkan nilai untuk hari 1.
getDay2()		Mengambil nilai jadwal untuk hari 2
setDay2(String day2)	Day2 (String)	Menetapkan nilai untuk hari 2
getDay3()		Mengambil nilai jadwal untuk hari 3
setDay3(String day3)	Day3 (String)	Menetapkan nilai untuk hari 3
getDay31()		Mengambil nilai jadwal untuk hari 31
setDay31(String day31)	Day31 (String)	Menetapkan nilai untuk hari 31

Pada kelas schedule, terdapat 10 method yang telah kami identifikasi. Berikut ini penjelasan untuk masing-masing metode tersebut.

THANK YOU FOR ATTENDING

Jika ingin melihat proyek kami lebih dalam,
dapat mengakses link berikut.

Our Documentation



[Link Github 15_Sistem Pengelolaan Event Organizer](#)



[Link G-Drive 15_Sistem Pengelolaan Event Organizer](#)

