**Laporan Tugas Proyek**

**Pemrograman Berorientasi Objek**

**Sistem Pengelolaan Event Organizer**



**Disusun Oleh :**

|  |  |
| --- | --- |
| 11323010 | Frangklyn Aldo Ignatia Lumbantoruan |
| 11323020 | Marshanda Kasih Simangunsong |
| 11323055 | Febyanti Hutahaean |

**Institut Teknologi Del**

**Fakultas Vokasi**

**DIII- Teknologi Informasi**

**Laguboti**

**2024**

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc185545226)

[DAFTAR GAMBAR 3](#_Toc185545227)

[DAFTAR TABEL 4](#_Toc185545228)

[1 Pendahuluan 5](#_Toc185545229)

[1.1 Latar belakang 5](#_Toc185545230)

[1.2 Rumusan masalah 5](#_Toc185545231)

[1.3 Tujuan 5](#_Toc185545232)

[1.4 Manfaat 6](#_Toc185545233)

[1.4.1 Manfaat bagi EO: 6](#_Toc185545234)

[1.4.2 Manfaat bagi mahasiswa pengembang: 6](#_Toc185545235)

[2 Desain dan Struktur Sistem 7](#_Toc185545236)

[2.1 Deskripsi proyek 7](#_Toc185545237)

[2.2 Fitur utama 7](#_Toc185545238)

[2.3 ER-Diagram 7](#_Toc185545239)

[2.4 Bisnis Proses Sistem 8](#_Toc185545240)

[2.5 Struktur kelas 8](#_Toc185545241)

[2.5.1 Struktur ClassLoginContoller 8](#_Toc185545242)

[2.5.2 Struktur Class RegisterController 9](#_Toc185545243)

[2.5.3 Struktur Class ClientContreller 11](#_Toc185545244)

[2.5.4 Struktur Class EventController 12](#_Toc185545245)

[2.5.5 Struktur Class ScheduleController 12](#_Toc185545246)

[2.5.6 Struktur Class MainController 13](#_Toc185545247)

[2.5.7 Struktur Class Client 14](#_Toc185545248)

[2.5.8 Struktur Class Event 15](#_Toc185545249)

[2.5.9 Struktur Class Schedule 16](#_Toc185545250)

[3 Penerapan Konsep OOP 18](#_Toc185545251)

[3.1 Penerapan OOP pada Class clientContreller 18](#_Toc185545252)

[3.2 Penerapan OOP pada Class cventContreller 20](#_Toc185545253)

[3.3 Penerapan OOP pada Class loginContreller 22](#_Toc185545254)

[3.4 Penerapan OOP Pada cLass MainContreeler 23](#_Toc185545255)

[3.5 Penerapan OOP pada class register controller 25](#_Toc185545256)

[3.6 Penerapan oop pada classs ScheduleController 27](#_Toc185545257)

[3.7 Penerapan oop pada class client 29](#_Toc185545258)

[3.8 Penerapan oop pada class event 30](#_Toc185545259)

[3.9 Penerapan oop pada class schedule 31](#_Toc185545260)

[4 Hasil Implementasi 32](#_Toc185545261)

[4.1 Hasil implementasi register 32](#_Toc185545262)

[4.2 Hasil Implementasi Login 33](#_Toc185545263)

[4.3 Hasil Implementasi dashboard 34](#_Toc185545264)

[4.4 Hasil Implementasi client 35](#_Toc185545265)

[4.4.1 Hasil implementasi tambah client 36](#_Toc185545266)

[4.4.2 Hasil implementasi edit client 37](#_Toc185545267)

[4.4.3 Hasil implementasi hapus client 38](#_Toc185545268)

[4.5 Hasil Implementasi Event 39](#_Toc185545269)

[4.6 Hasil Implementasi Schedule 40](#_Toc185545270)

[5 Kesimpulan Dan Saran 42](#_Toc185545271)

[5.1 Kesimpulan 42](#_Toc185545272)

[6 Link YT Presentasi 43](#_Toc185545273)

# DAFTAR GAMBAR

[Gambar 1. Er-Diagram Sistem Pengelolaan Event Organizer 7](#_Toc185545274)

[Gambar 2. Bisni Proses Sistem Pengelolaan Event Organizer 8](#_Toc185545275)

[Gambar 3 Abstraction Pada Class Clientcontreller 18](#_Toc185545276)

[Gambar 4 Encapsulation Pada Class Clientcontreller 19](#_Toc185545277)

[Gambar 5 Inheritance Class Clientcontreller 19](#_Toc185545278)

[Gambar 6 Polumorphism Class Clientcontreller 19](#_Toc185545279)

[Gambar 7 Exception Handling Pada Class Clientcontreller 20](#_Toc185545280)

[Gambar 8 Abstraction Pada Class Eventcontreller 20](#_Toc185545281)

[Gambar 9 Encapsulation Pada Class Eventcontreller 21](#_Toc185545282)

[Gambar 10 Inheritance Pada Class Ecentcontroller 21](#_Toc185545283)

[Gambar 11 Polymorphism Pada Class Eventcontreller 22](#_Toc185545284)

[Gambar 12 Encapsulation Pada Class Logincontroller 22](#_Toc185545285)

[Gambar 13 Polymorphism Pada Class Logincontroller 23](#_Toc185545286)

[Gambar 14 Abstraction Pada Class Logincontroller 23](#_Toc185545287)

[Gambar 15 Encapsulation Pada Maincontrllor 24](#_Toc185545288)

[Gambar 16 Abstraction Pada Cllass Maincontroller 24](#_Toc185545289)

[Gambar 17 Exception Handling Pada Class Maincontroller 25](#_Toc185545290)

[Gambar 18 Encapsulation Pada Class Registercontroller 26](#_Toc185545291)

[Gambar 19 Abstraction Pada Class Registercontroller 26](#_Toc185545292)

[Gambar 20 Exception Handling Pada Class Reistercontroller 26](#_Toc185545293)

[Gambar 21 Encapsulation Pada Class Schedulecontoller 27](#_Toc185545294)

[Gambar 22 Polymorphism Pada Class Schedulecontroller 27](#_Toc185545295)

[Gambar 23 Abstraction Pada Class Schedulecontroller 28](#_Toc185545296)

[Gambar 24 Interface Pada Class Schedulecontroller 28](#_Toc185545297)

[Gambar 25 Encapsulation Pada Class Client 29](#_Toc185545298)

[Gambar 26 Encapsulation Pada Class Event 30](#_Toc185545299)

[Gambar 27 Encapsulation Pada Class Schedule 31](#_Toc185545300)

[Gambar 28 Hasil Implementasi Register 32](#_Toc185545301)

[Gambar 29 Hasil Implemetasi Register Jika Tidak Seusal Kriteria 32](#_Toc185545302)

[Gambar 30 Hasil Implementasi Register Jika Suda Berhasil Buat Akun 33](#_Toc185545303)

[Gambar 31 Hasil Implementasi Login 33](#_Toc185545304)

[Gambar 32 Hasil Implementasi Login Ketika Berhasil Masuk 34](#_Toc185545305)

[Gambar 33 Hasil Implementasi Dashboard Jika Semua Event Selesai 34](#_Toc185545306)

[Gambar 34 Hasil Implementasi Dashboard Jika Sebagian Event Belum Selesai 35](#_Toc185545307)

[Gambar 35 Hasial Implementasi Client 35](#_Toc185545308)

[Gambar 36 Hasil Implementasi Tambah Client 36](#_Toc185545309)

[Gambar 37 Hasil Implementasi Jika Client Berhasil Ditambah 36](#_Toc185545310)

[Gambar 38 Hasil Implementasi Edit Client 37](#_Toc185545311)

[Gambar 39 Hasil Implementasi Jika Client Berhasil Di Edit 37](#_Toc185545312)

[Gambar 40 Hasil Implementasi Hapus Client 38](#_Toc185545313)

[Gambar 41 Hasil Implementasi Hapus Client 38](#_Toc185545314)

[Gambar 42 Implementasi Tambah Event 39](#_Toc185545315)

[Gambar 43 Hasil Implementasi Tambah Event 39](#_Toc185545316)

[Gambar 44 Implementasi Tambah Schedule 40](#_Toc185545317)

[Gambar 45 Hasil Impementasi Tambah Schedule 40](#_Toc185545318)

[Gambar 46 Implementasi Jika Tidak Ada Lagi Event Yang Ingin Ditambahakan 41](#_Toc185545319)

# DAFTAR TABEL

[tabel 1 method ClassLoginContoller 9](#_Toc185545328)

[tabel 2 Method RegisterController 11](#_Toc185545329)

[tabel 3 Method Class ClientContreller 11](#_Toc185545330)

[tabel 4 method class EventController 12](#_Toc185545331)

[tabel 5 method ScheduleController 13](#_Toc185545332)

[tabel 6 method MainController 14](#_Toc185545333)

[tabel 7 method Class Client 15](#_Toc185545334)

[tabel 8 method Class Event 16](#_Toc185545335)

[tabel 9 method Class Schedule 17](#_Toc185545336)

# Pendahuluan

## Latar belakang

Event Organizer (EO) memiliki peran penting dalam mengelola berbagai jenis acara, mulai dari perencanaan hingga pelaksanaan. Acara seperti pernikahan, seminar, konser, dan konferensi sering kali melibatkan banyak pihak yang memerlukan koordinasi yang baik agar acara berjalan lancar. Proses pengelolaan ini semakin kompleks seiring dengan bertambahnya jumlah klien dan acara yang harus ditangani secara bersamaan

Berbagai aspek seperti pengelolaan data klien, informasi acara, hingga penjadwalan membutuhkan perhatian khusus untuk menghindari kesalahan yang dapat memengaruhi kualitas layanan EO. Oleh karena itu, sistem pengelolaan berbasis aplikasi menjadi solusi yang dapat Ru perhatian khusus untuk menghindari kesalahan yang dapat memengaruhi kualitas layanan EO. Oleh karena itu, sistem pengelolaan berbasis aplikasi menjadi solusi yang dapat membantu EO dalam meningkatkan efisiensi, akurasi, dan efektivitas operasional mereka

## Rumusan masalah

**Rumusan masalah** berikut dirancang untuk mengidentifikasi tantangan utama yang dihadapi dalam pengelolaan data dan operasional Event Organizer. Beberapa permasalahan yang perlu diselesaikan adalah sebagai berikut:

1. Bagaimana cara mempermudah pengelolaan data klien oleh Event Organizer?
2. Bagaimana memastikan informasi acara yang terorganisir dan mudah diakses?
3. Bagaimana membuat jadwal acara yang dapat dipantau statusnya (aktif/selesi)?

## Tujuan

Dalam pengembangan sistem ini, terdapat beberapa tujuan utama yang ingin dicapai untuk mendukung operasional Event Organizer secara lebih efektif. Tujuan tersebut meliputi :

1. Mengembangkan aplikasi berbasis teknologi untuk membantu EO dalam mengelola data klien, informasi acara, dan jadwal secara terpusat.
2. Menyediakan fitur yang intuitif dan mudah digunakan untuk meningkatkan pengalaman pengguna.
3. Meningkatkan efisiensi dan akurasi dalam proses pengelolaan data EO.
4. Mempermudah pemantauan status acara (aktif/selesai) secara real-time.

## Manfaat

Pengembangan sistem ini memberikan manfaat tidak hanya bagi Event Organizer (EO) sebagai pengguna utama, tetapi juga bagi mahasiswa sebagai pengembang sistem. Berikut adalah manfaat yang diharapkan:

### Manfaat bagi EO:

1. Memudahkan EO dalam menyimpan dan mengakses informasi klien serta acara tanpa kendala.
2. Mengurangi potensi kesalahan dalam pengelolaan data yang dilakukan secara manual, sehingga meningkatkan keandalan sistem.
3. Meningkatkan produktivitas tim EO dengan menyediakan sistem yang terintegrasi dan dapat diakses kapan saja.
4. Meningkatkan kepuasan klien dengan layanan yang lebih cepat, terorganisir, dan profesional.
5. Memberikan alat bantu analitik untuk memahami kebutuhan klien dan tren acara, yang dapat membantu pengambilan keputusan strategis.

### Manfaat bagi mahasiswa pengembang:

1. Memberikan pengalaman langsung dalam menerapkan konsep Pemrograman Berorientasi Objek (PBO) ke dalam pengembangan aplikasi nyata.
2. Meningkatkan pemahaman terhadap siklus pengembangan perangkat lunak, mulai dari analisis kebutuhan hingga implementasi.
3. Melatih keterampilan teknis dalam merancang dan mengintegrasikan sistem berbasis teknologi untuk menyelesaikan permasalahan dunia nyata.
4. Menumbuhkan kemampuan kolaborasi dan manajemen proyek, karena pengembangan melibatkan diskusi dan pembagian tugas.
5. Menjadi portofolio berharga sebagai bukti kemampuan mahasiswa dalam mengembangkan solusi inovatif yang aplikatif.

# Desain dan Struktur Sistem

## Deskripsi proyek

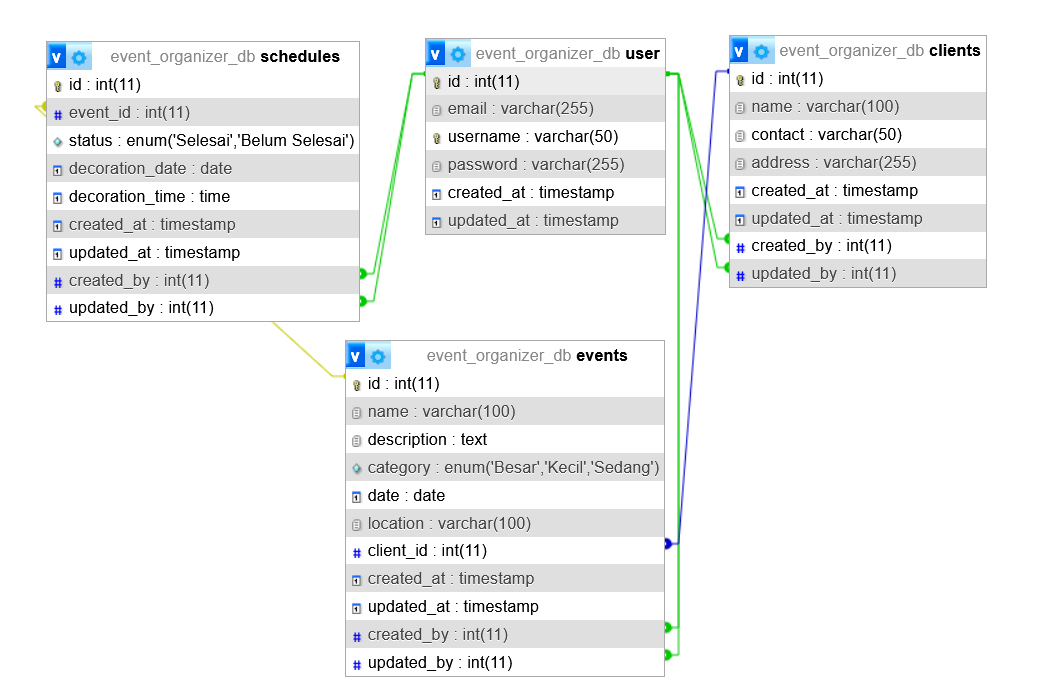
Sistem Pengelolaan Event Organizer (EO) adalah sistem yang dirancang untuk mempermudah pengelolaan data terkait klien, acara, dan jadwal acara secara terpusat. Sistem ini akan dilengkapi dengan antarmuka yang user-friendly dan fitur yang komprehensif untuk mendukung kebutuhan operasional EO.

## Fitur utama

Berikut adalah fitur utama yang dikembangkan dalam sistem :

1. **Register: Fitur yang memungkinkan pengguna baru untuk membuat akun dan mendaftar ke dalam sistem dengan memasukkan informasi yang diperlukan.**
2. **Login: Memungkinkan pengguna yang sudah terdaftar untuk masuk ke dalam sistem dengan menggunakan kredensial yang valid (username dan password) guna mengakses fitur-fitur yang tersedia.**
3. **Manajemen Klien: Memungkinkan EO untuk menambahkan, mengubah, dan menghapus data klien, serta menyimpan detail acara yang terkait dengan masing-masing klien.**
4. **Manajemen Acara: Mengelola informasi acara seperti jenis acara, tanggal pelaksanaan, dan lokasi secara efisien.**
5. **Pengelolaan Jadwal Acara: Memberikan kemampuan untuk memantau status acara (aktif atau selesai) dan mencari acara berdasarkan tanggal tertentu.**

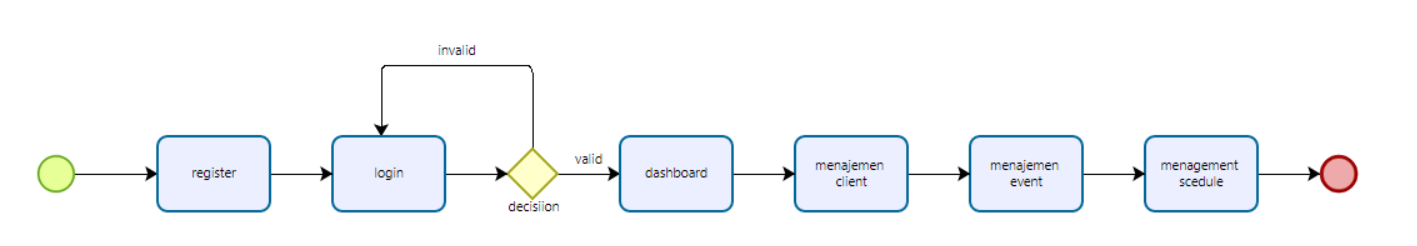
## ****ER-Diagram****

****

Gambar 1. ER-Diagram Sistem Pengelolaan Event Organizer

Pada ER-Diagram ini, terlihat hubungan antara beberapa entitas yang membentuk sistem pengelolaan Event Organizer. Entitas **user** menyimpan data pengguna yang mengelola sistem, dengan relasi ke entitas **clients**, **events**, dan **schedules** melalui atribut created\_by dan updated\_by, yang menunjukkan siapa yang membuat atau memperbarui data tersebut. Entitas **clients** menyimpan data klien, yang memiliki hubungan **one-to-many** dengan entitas **events**, dimana satu klien dapat memiliki banyak acara. Setiap acara memiliki relasi dengan entitas **schedules**, yang menyimpan jadwal terkait dengan acara tersebut, termasuk status dan waktu dekorasi. Dengan struktur ini, sistem dapat secara efektif mengelola data klien, acara, dan jadwal secara terpusat, serta memastikan interaksi antar entitas berjalan dengan baik.

## Bisnis Proses Sistem

Gambar 2. Bisni Proses Sistem Pengelolaan Event Organizer

Gambar 2 menggambarkan alur bisnis proses sistem pengolahan event organizer yang dimulai dengan registrasi pengguna, diikuti dengan proses login untuk validasi akses. Jika login berhasil, pengguna diarahkan ke dashboard sebagai pusat pengelolaan, di mana mereka dapat mengelola data klien dan event secara terstruktur. Tahapan berikutnya melibatkan pengelolaan solusi manajemen untuk menyusun rencana dan eksekusi event secara menyeluruh. Proses ini memastikan semua aktivitas dari perencanaan hingga penyelesaian event dapat dilakukan dengan efisien dan terorganisir.

## Struktur kelas

### Struktur ClassLoginContoller

* Deskripso

LoginController adalah **controller class** yang bertanggung jawab untuk mengatur logika proses login pada aplikasi berbasis JavaFX. Class ini bertindak sebagai penghubung antara tampilan (View) dan logika (Model) dalam pola arsitektur **MVC (Model-View-Controller)**. Class ini mengatur pengambilan data dari form login, memvalidasi input, memproses autentikasi dengan basis data, dan mengalihkan ke halaman yang sesuai (seperti halaman utama atau halaman registrasi).

* Fungsi:
  1. Mengelola Proses Login: Mengambil data dari input pengguna, memvalidasinya, dan mengecek ke basis data apakah username dan password cocok.
  2. Navigasi Halaman: Berpindah dari halaman login ke halaman utama (MainView) atau halaman registrasi (Register).
  3. Memberikan Notifikasi: Menampilkan pesan notifikasi sementara kepada pengguna, baik saat login berhasil, gagal, atau jika ada kesalahan input.
* Method

|  |  |  |
| --- | --- | --- |
| 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] |

tabel 1 method ClassLoginContoller

### Struktur Class RegisterController

* Deskripsi

Class RegisterController adalah bagian dari arsitektur **MVC (Model-View-Controller)**. Class ini bertanggung jawab untuk menangani proses pendaftaran pengguna pada aplikasi Event Organizer. Proses ini mencakup pengambilan data input dari pengguna, validasi data input, pengecekan ketersediaan username, penyimpanan data ke database, serta navigasi ke halaman login setelah pendaftaran berhasil.

* Fungsi

Fungsi utama dari class RegisterController adalah sebagai berikut:

1. Mengelola Input Pendaftaran: Mengambil input dari TextField (username, email, dan password) dan memvalidasi input tersebut.
2. Validasi Data: Memastikan bahwa username unik, email valid (mengandung @gmail.com), dan password memenuhi kriteria minimal (6 karakter, mengandung huruf dan angka).
3. Menyimpan Data ke Database: Menyimpan informasi pengguna ke dalam tabel **user** di database jika semua validasi berhasil.
4. Navigasi Halaman: Mengarahkan pengguna ke halaman **Login** setelah pendaftaran berhasil.
5. Menampilkan Notifikasi dan Alert: Memberikan feedback kepada pengguna, baik dalam bentuk notifikasi sementara maupun alert dialog.

* Merhod

|  |  |  |
| --- | --- | --- |
| Nama Metode | 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. |

tabel 2 Method RegisterController

### Struktur Class ClientContreller

* Deskripsi:

Class ini berperan sebagai **pengendali (controller)** yang mengatur interaksi antara antarmuka pengguna (UI) dan logika aplikasi. Class ini bertanggung jawab menangani input dari pengguna dan mengatur data yang akan ditampilkan pada tabel.

* Fungsi:
  1. Menangani interaksi UI (klik tombol, pengisian form).
  2. Mengelola form popup untuk menambah data client baru.
  3. Menampilkan daftar client di dalam **TableView**.
* Method:

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

tabel 3 Method Class ClientContreller

### Struktur Class EventController

* Deskripsi  
  Kelas ini adalah **controller** dalam aplikasi berbasis JavaFX. Kelas ini bertanggung jawab untuk mengelola interaksi antara tampilan (view) dan data (model) pada tabel event. Kelas ini mengatur data dari database MySQL ke tampilan tabel.
* Fungsi
  1. Mengelola data event dari database MySQL
  2. menampilkannya di UI (TableView). Selain itu, mengatur interaksi user seperti mengisi tabel secara otomatis dari database.
* Method

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

tabel 4 method class EventController

### Struktur Class ScheduleController

* Deskripsi

Kelas ini adalah **controller** dalam aplikasi berbasis JavaFX. Kelas ini bertanggung jawab untuk mengatur interaksi antara tampilan (view) dan data.

* Fungsi

1. Mengelola tampilan dan logika interaksi pengguna di halaman jadwal
2. Mengatur tampilan popup dan tabel jadwal.

* Method

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

tabel 5 method ScheduleController

### Struktur Class MainController

* Deskripsi

Kelas ini bertanggung jawab untuk mengelola interaksi antara tampilan (view) dan logika aplikasi (business logic) terkait statistik dan data event.

* Funsgi

1. Mengelola tampilan statistik jumlah pengguna, klien, dan event,
2. menampilkan grafik status event dan data klien dalam tabel.

* Method

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

tabel 6 method MainController

### Struktur Class Client

* Deskripsi

Kelas ini bertanggung jawab untuk mengelola interaksi antara tampilan (view) dan logika aplikasi (business logic) terkait statistik dan data event.

* Funsgi

1. Menyimpan data client (nama, kontak, dan alamat).
2. Menyediakan **getter** dan **setter** agar data dapat diakses dan diperbarui.
3. Mempermudah pengelolaan data client di **TableView** aplikasi.

* Method

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

tabel 7 method Class Client

### Struktur Class Event

* Deskripsi

Kelas ini berfungsi untuk mengatur interaksi antara tampilan pengguna (UI) dan logika aplikasi terkait jadwal. Kelas ini menangani pengaturan tampilan tabel jadwal (TableView) dan popup untuk menambahkan jadwal baru.

* Funsgi
  + 1. Mengelola tampilan dan pengaturan elemen UI yang terkait dengan jadwal.Mengatur visibilitas popup yang digunakan untuk menambahkan jadwal.
    2. Mengelola data yang ditampilkan dalam TableView.
* Method

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

tabel 8 method Class Event

### Struktur Class Schedule

* Deskripsi

Kelas Schedule digunakan untuk menyimpan jadwal yang terdiri dari jam dan kegiatan atau acara untuk setiap hari dalam sebulan. Setiap hari (dari hari 1 hingga hari 31) memiliki atribut sendiri yang berisi informasi terkait jadwal yang dilakukan pada hari tersebut. Kelas ini memiliki getter dan setter untuk setiap atribut, yang memungkinkan untuk mendapatkan dan mengatur jadwal berdasarkan jam dan hari tertentu.

* Funsgi:
  + 1. **Menyimpan Jadwal**: Kelas ini berfungsi untuk menyimpan informasi jadwal berdasarkan jam dan hari dalam sebulan.
    2. **Mengakses dan Memodifikasi Data Jadwal**: Dengan menggunakan getter dan setter, kelas ini memungkinkan kita untuk mengakses jadwal per hari atau mengubah jadwal yang telah ditentukan sebelumnya.
* Method

|  |  |  |
| --- | --- | --- |
| 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 |

tabel 9 method Class Schedule

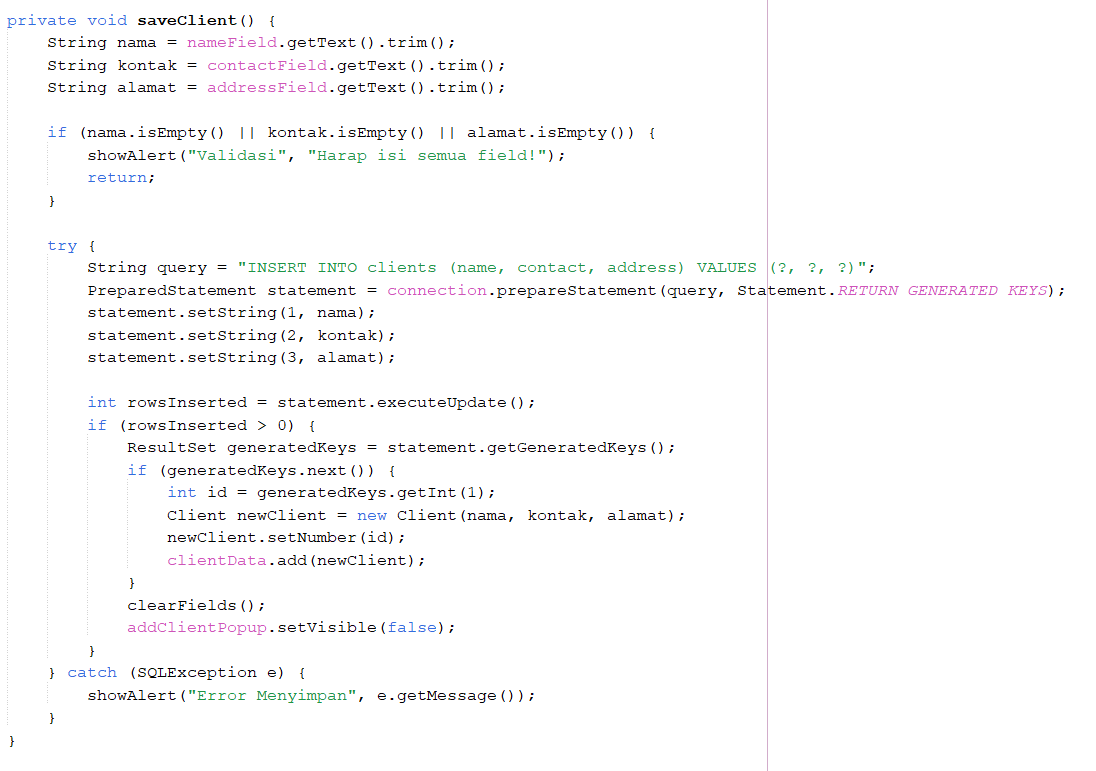
# Penerapan Konsep OOP

## Penerapan OOP pada Class clientContreller

Pada class ini terdapat penerapan beberapa konsep OOP, yaitu **Abstraction**, **Encapsulation, Inheritance, Polymorphism, Interface,** dan **Exception Handling**. Berikut penjelasannya:

* Abstraction

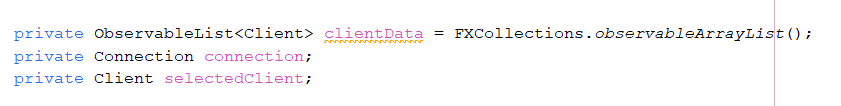
Abstraksi adalah konsep untuk menyembunyikan kompleksitas implementasi dan hanya menampilkan fungsionalitas yang diperlukan kepada pengguna. Pada kode ini, kelas ClientController menyembunyikan banyak detail implementasi terkait dengan database dan antarmuka pengguna. Misalnya, cara menyimpan, memperbarui, dan menghapus data client disembunyikan dalam metode-metode seperti saveClient(), updateClient(), dan deleteClient()



Gambar 3 Abstraction pada Class clientContreller

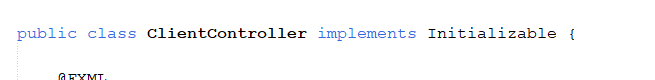
* Encapsulation

Konsep encapsulation diterapkan dengan cara menyembunyikan data menggunakan **modifier akses** private pada atribut dan hanya memungkinkan akses melalui metode **getter dan setter**.contoh nya pada Atribut seperti connection, clientData, dan selectedClient dideklarasikan sebagai private, sehingga tidak dapat diakses secara langsung dari luar kelas. Akses hanya dapat dilakukan melalui metode atau proses internal kelas tersebut.

**Gambar 4 encapsulation pada Class clientContreller**

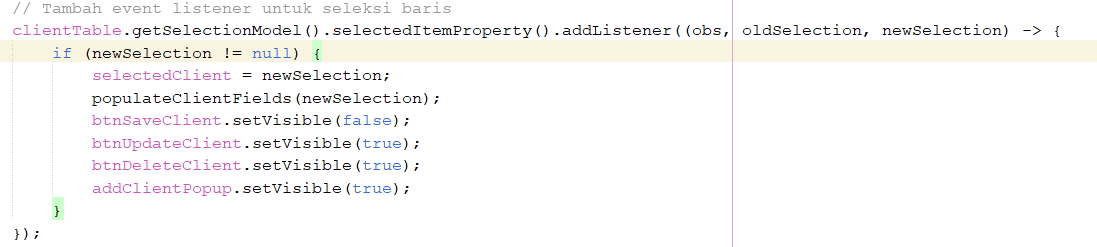
* Inheritance

ClientController mewarisi dari Initializable. Dengan mewarisi Initializable, kelas ini diwajibkan untuk mengimplementasikan metode initialize(). Metode ini dipanggil secara otomatis saat file FXML dimuat.

**Gambar 5 Inheritance Class clientContreller**

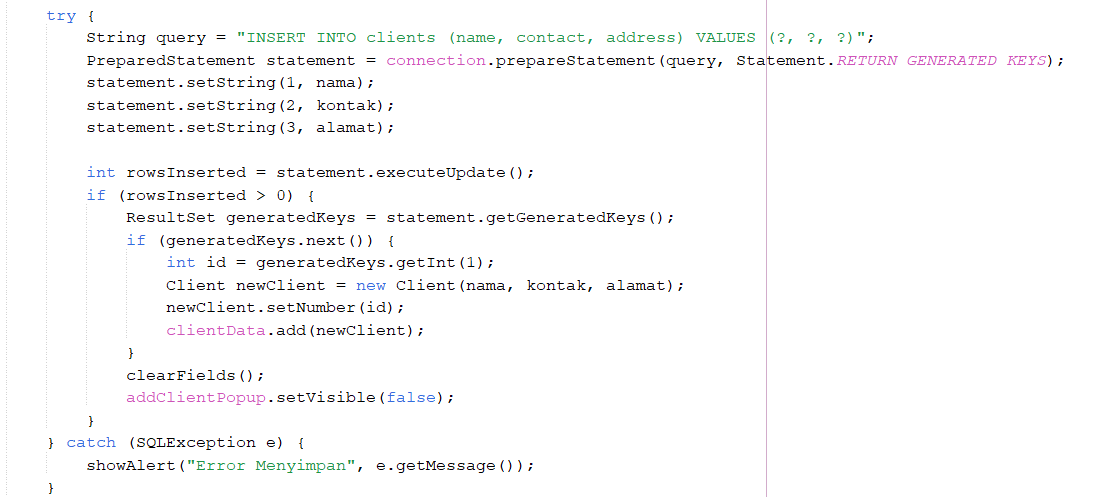
* Polymorphism

Konsep polymorphism diterapkan dengan memungkinkan metode yang sama memiliki perilaku yang berbeda pada situasi tertentu. Listener pada TableView: clientTable.getSelectionModel().selectedItemProperty().addListener() menggunakan Lambda Expression yang merupakan salah satu bentuk dari polimorfisme runtime. Saat baris pada tabel dipilih, objek Client yang dipilih dapat memicu perilaku berbeda tergantung data dari baris tersebut.

**Gambar 6 polumorphism Class clientContreller**

* Exception Handling

Pada metode goToClient(), proses pemuatan file FXML dapat gagal karena file mungkin hilang atau rusak. Untuk menghindari error yang tidak terduga, digunakan try-catch untuk menangkap dan menampilkan pesan kesalahan.  
Exception handling juga diterapkan pada operasi database, seperti insert, update, dan delete data klien, menggunakan try-catch untuk menangkap SQLException.

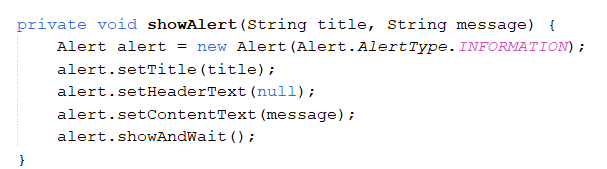
**Gambar 7 Exception Handling pada Class clientContreller**

## Penerapan OOP pada Class cventContreller

Pada class ini terdapat penerapan beberapa konsep OOP, yaitu **Abstraction**, **Encapsulation, Inheritance, Polymorphism,** . Berikut penjelasannya:

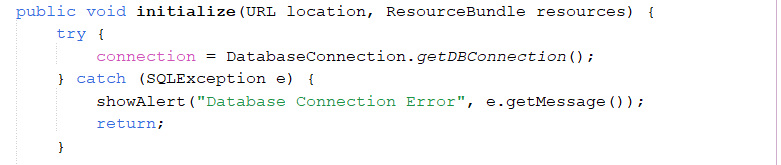
* Abstraction

Metode showAlert() adalah contoh abstraction. Metode ini menyembunyikan detail pembuatan dan pengaturan Alert dari pengguna. Pengguna hanya perlu memanggil showAlert() dengan judul dan pesan, tanpa perlu tahu bagaimana detail Alert tersebut dibuat.

**Gambar 8 abstraction pada Class eventContreller**

* Encapsulation

Encapsulation adalah pembungkusan data (atribut) dan metode dalam satu unit (kelas) serta membatasi aksesnya menggunakan akses modifier seperti private. Pada kode di atas, encapsulation diterapkan melalui atribut connection dan eventData, yang hanya dapat diakses melalui metode tertentu, bukan secara langsung.

**Gambar 9 Encapsulation pada class eventContreller**

* Inheritance

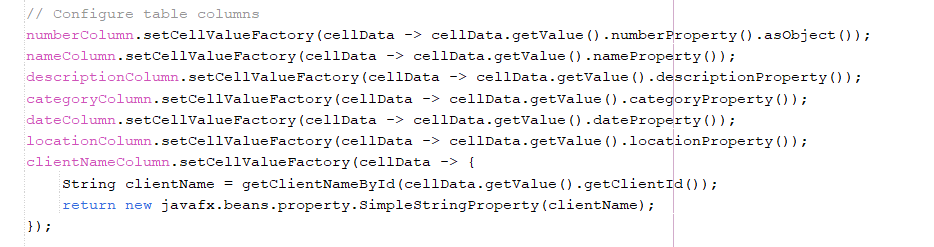
nheritance adalah konsep di mana sebuah kelas dapat mewarisi atribut dan metode dari kelas induknya. Dalam kode ini, EventController mewarisi dari Initializable, memungkinkan penggunaan metode initialize().



**Gambar 10 inheritance pada class ecentcontroller**

* Polymorpism

Polymorphism memungkinkan satu metode memiliki banyak bentuk. Dalam kode ini, polymorphism diterapkan melalui metode setCellValueFactory(), yang dapat **mengambil berbagai jenis data properti** (seperti Integer, String, dan Date) di kolom tabel.

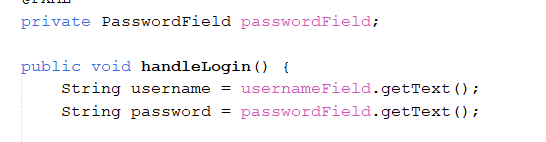
**Gambar 11 polymorphism pada class eventContreller**

## Penerapan OOP pada Class loginContreller

Pada class ini terdapat penerapan beberapa konsep OOP, yaitu **Abstraction**, **Encapsulation, Polymorphism,** Berikut penjelasannya:

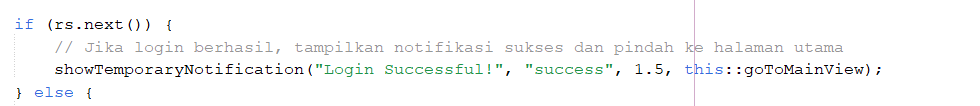
* Encapsulation

Encapsulation adalah konsep yang menyembunyikan data internal dan hanya memperbolehkan akses melalui metode tertentu. Pada kode ini, penggunaan usernameField dan passwordField adalah contoh dari encapsulation, karena mereka bersifat private dan hanya bisa diakses atau dimodifikasi melalui metode yang disediakan.

**Gambar 12 encapsulation pada class loginController**

* Polymorphism

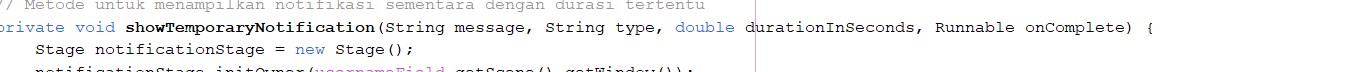
Polymorphism memungkinkan satu metode untuk bekerja dengan berbagai tipe objek. Pada kode ini, polymorphism tidak terlalu eksplisit, namun dapat dilihat dari metode showTemporaryNotification() yang menerima parameter Runnable onComplete, yang bisa berbeda-beda implementasinya, tergantung pada apa yang ingin dilakukan setelah notifikasi selesai.

\

**Gambar 13 polymorphism pada class logincontroller**

* Abstraction

Abstraction adalah konsep yang menyembunyikan implementasi internal dan hanya menyediakan interface yang diperlukan. Metode showTemporaryNotification() adalah contoh abstraction, di mana detail tentang bagaimana notifikasi ditampilkan disembunyikan dari pengguna, dan mereka hanya perlu memberikan parameter seperti pesan, tipe, durasi, dan aksi lanjutan.

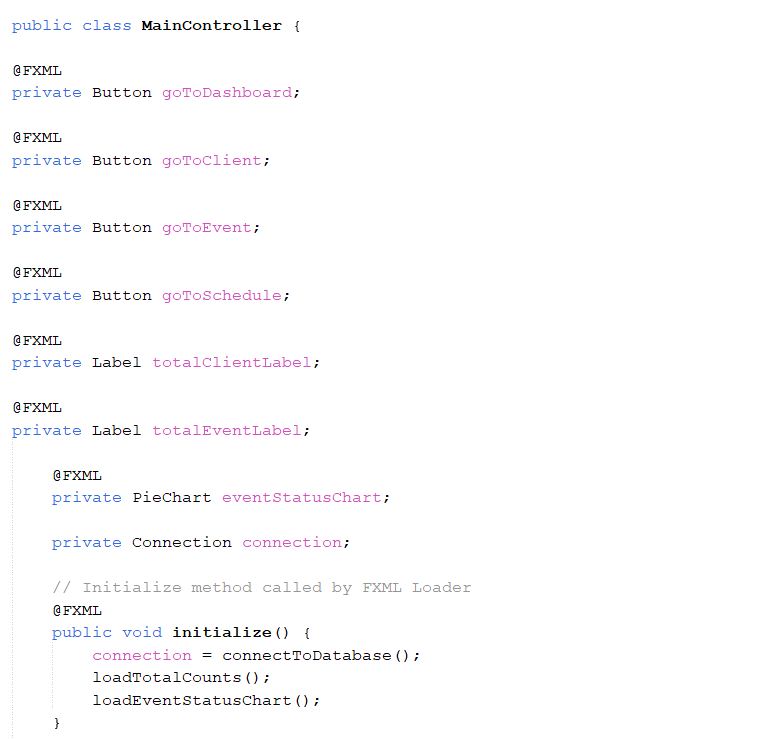
**Gambar 14 abstraction pada class logincontroller**

## Penerapan OOP Pada cLass MainContreeler

Pada class ini terdapat penerapan beberapa konsep OOP, yaitu **Abstraction**, **Encapsulation, Polymorphism, resubality,** **Exception Handling ,** Berikut penjelasannya:

* Encapsulation

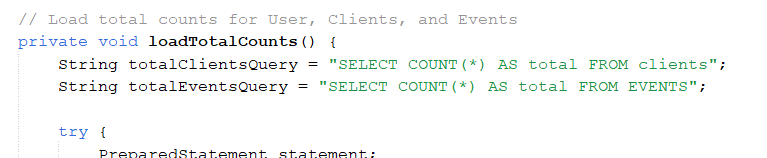
Metode connectToDatabase() dan loadTotalCounts() menyembunyikan detail implementasi terkait database dari kode lain. Hanya metode-metode ini yang mengakses data dan mengatur UI sesuai kebutuhan.



**Gambar 15 encapsulation pada maincontrllor**

* Abstraction

loadTotalCounts() dan loadEventStatusChart() adalah contoh penggunaan abstraksi. Mereka mengabstraksi logika yang kompleks seperti query database dan pembaruan UI agar lebih mudah digunakan tanpa memikirkan detail implementasi.

**Gambar 16 abstraction pada cllass maincontroller**

* **Exception Handling**:

Dalam setiap metode yang melibatkan operasi database, error handling menggunakan try-catch untuk menangani potensi masalah dengan koneksi atau query, yang memperlihatkan bagaimana error dikelola dengan baik dalam OOP.



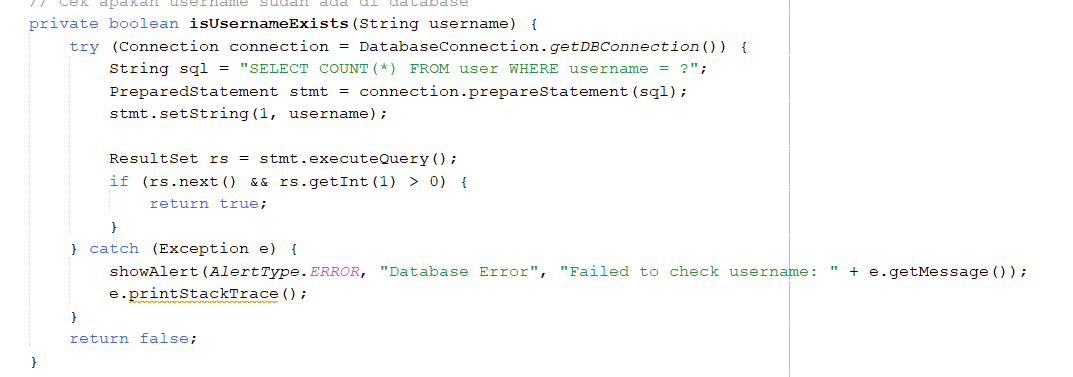
**Gambar 17 Exception Handling pada class maincontroller**

## Penerapan OOP pada class register controller

Pada class ini terdapat penerapan beberapa konsep OOP, yaitu **Abstraction**, **Encapsulation, Polymorphism, resubality,** **Exception Handling ,** Berikut penjelasannya:

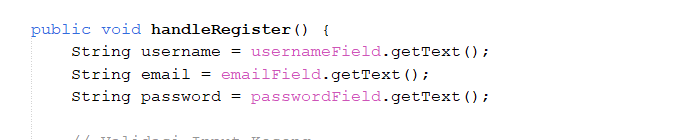
* **Encapsulation**

Enkapsulasi diterapkan di sini dengan cara menyembunyikan detail implementasi, seperti proses validasi dan interaksi dengan database, dan hanya menyediakan antarmuka (metode) untuk digunakan oleh pengguna kelas. Contohnya adalah penggunaan metode handleRegister(), yang menangani semua logika pendaftaran pengguna dan memvalidasi input tanpa mengekspose detail implementasi kepada pengguna kelas lainnya.

**Gambar 18 encapsulation pada class registercontroller**

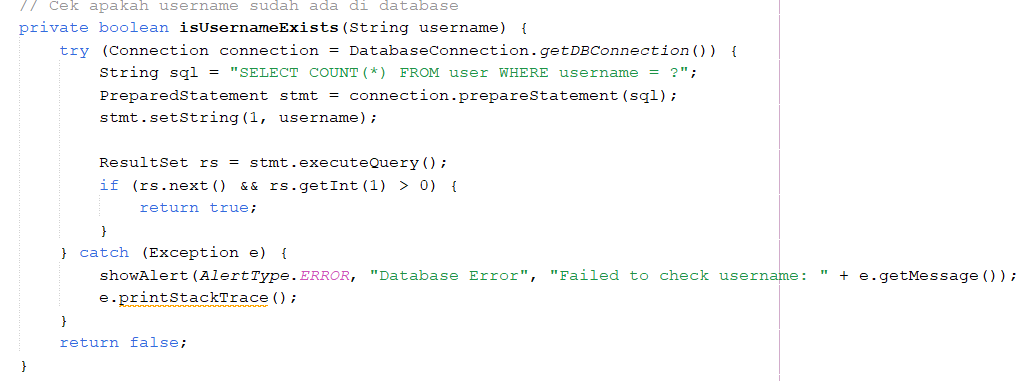
* Abstraction

Abstraksi diterapkan pada metode-metode seperti handleRegister() dan goToLogin(). Metode ini memberikan antarmuka yang jelas bagi pengguna kelas tanpa harus tahu bagaimana detail implementasinya.

**Gambar 19 abstraction pada class registercontroller**

* Exception Handling

Penanganan pengecualian diterapkan di seluruh metode yang berinteraksi dengan database, seperti isUsernameExists() dan handleRegister(). Jika terjadi kesalahan (misalnya kegagalan koneksi ke database), pengecualian akan ditangani dengan blok try-catch, dan pesan kesalahan akan ditampilkan kepada pengguna.

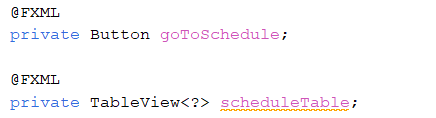
**Gambar 20 Exception Handling pada class reistercontroller**

## Penerapan oop pada classs ScheduleController

Pada class ini terdapat penerapan beberapa konsep OOP, yaitu **Abstraction**, **Encapsulation, Polymorphism, interface,** Berikut penjelasannya:

* Encapsulation

Encapsulation diterapkan dengan menyembunyikan detail implementasi elemen antarmuka pengguna seperti Button, TableView, dan AnchorPane. Elemen-elemen ini hanya diakses melalui metode di dalam kelas, menjadikan kode lebih terstruktur dan mudah dipelihara.



Gambar 21 encapsulation pada class scheduleContoller

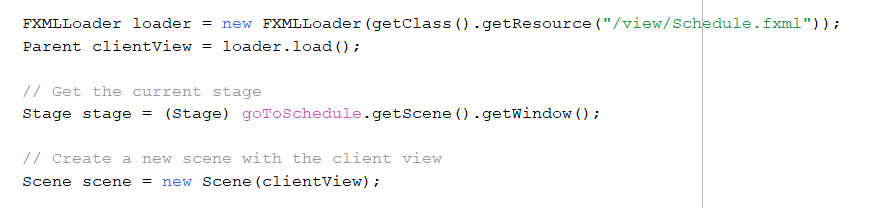
* Polymorphism

Polymorphism diterapkan melalui metode navigasi yang berbeda untuk setiap tombol (goToClient, goToDashboard, dll.). Setiap metode memuat tampilan yang berbeda, namun mereka memiliki nama yang serupa, menunjukkan kesamaan dalam fungsionalitas meskipun implementasinya berbeda.

**Gambar 22 polymorphism pada class schedulecontroller**

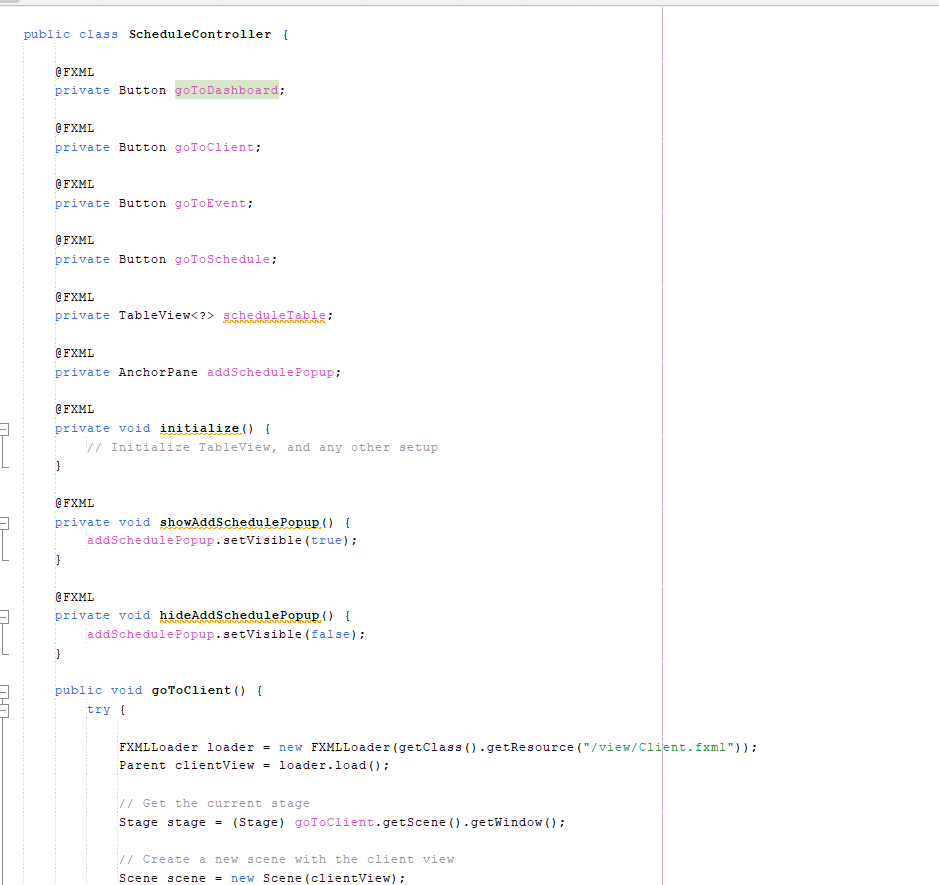
* Abstraction

Dengan menggunakan FXML dan Scene, rincian implementasi tampilan disembunyikan dari kode kontroler. Ini memungkinkan fokus pada logika aplikasi dan membuat tampilan dapat diganti tanpa mengubah logika navigasi.

**Gambar 23 Abstraction pada class schedulecontroller**

* Interface

elas ScheduleController bisa mengimplementasikan interface ini dan menyediakan implementasi untuk metode go to client. Ini memberikan fleksibilitas untuk menggunakan berbagai implementasi navigasi di berbagai controller.



**Gambar 24 Interface pada class scheduleController**

## Penerapan oop pada class client

Pada class ini terdapat penerapan konsep OOP, yaitu **Encapsulation,** Berikut penjelasannya:

* Encapsulation

Pada penerapan enkapsulasi, atribut dalam kelas Client disembunyikan dengan menggunakan modifier private, dan akses terhadap data hanya dapat dilakukan melalui metode **getter** dan **setter**. Dengan demikian, data hanya dapat dimodifikasi atau dibaca melalui cara yang telah ditentukan, menjaga integritas dan keamanan data.



**Gambar 25 Encapsulation pada class client**

## Penerapan oop pada class event

Pada class ini terdapat penerapan konsep OOP, yaitu **Encapsulation,** Berikut penjelasannya

* Encapsulation

Enkapsulasi adalah prinsip OOP yang menyembunyikan atribut atau data dalam kelas dan memberikan akses melalui metode khusus (getter dan setter). Dalam kelas Event, atribut seperti number, id, name, description, category, date, location, clientId, createdBy, dan updatedBy disembunyikan dengan menggunakan **private access modifier**. Akses ke data ini hanya dapat dilakukan melalui **getter** dan **setter**, yang memungkinkan pengontrolan dan perlindungan data.



**Gambar 26 encapsulation pada class event**

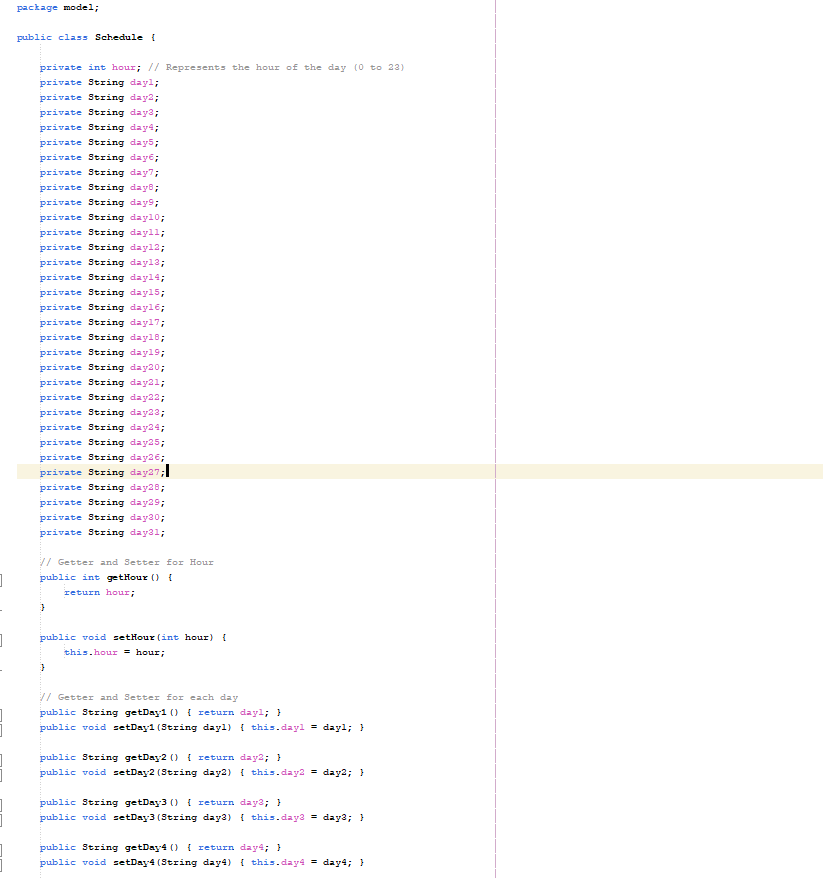
## Penerapan oop pada class schedule

Pada class ini terdapat penerapan konsep OOP, yaitu **Encapsulation,** Berikut penjelasannya

* Encapsulation

Encapsulation adalah prinsip OOP yang menyarankan untuk menyembunyikan data internal (atribut) dan hanya memberikan akses ke data tersebut melalui metode tertentu (getter dan setter). Hal ini bertujuan agar data tersebut tidak bisa diakses sembarangan, memberikan kontrol lebih terhadap perubahan nilai atribut.

Pada kelas **Schedule**, atribut seperti day1, day2, ..., day31 dan hour di-private-kan (tidak dapat diakses langsung dari luar kelas) dan hanya bisa diubah atau diambil nilainya melalui metode getter dan setter yang disediakan:

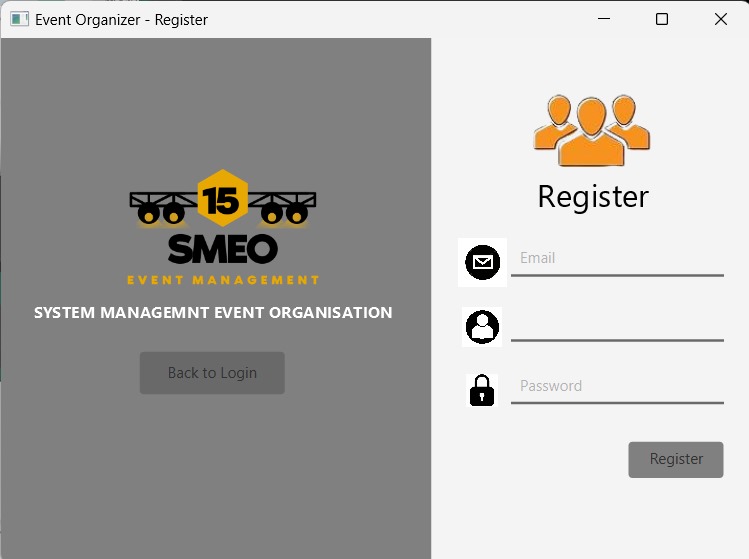


**Gambar 27 encapsulation pada class schedule**

# Hasil Implementasi

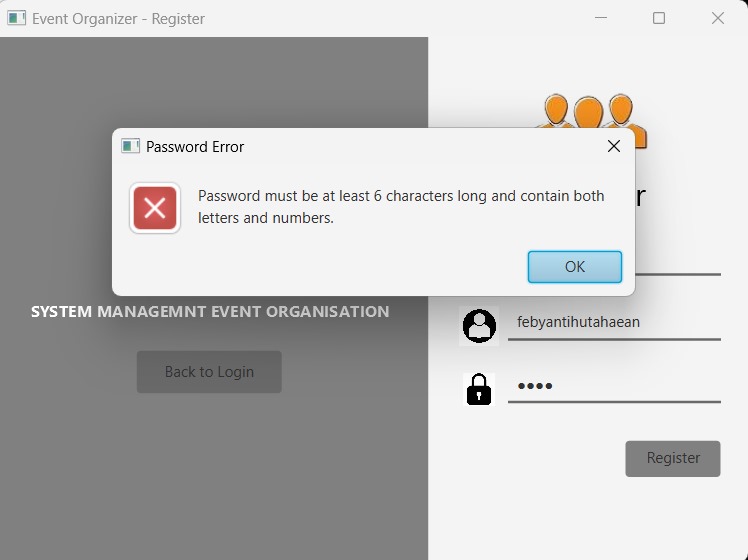
## Hasil implementasi register

Pada halaman ini, pengguna harus mengisi form Gmail, username dan password, setelah itu klik button register.Akan tampil raise notice berisikan akun berhasil didaftarkan.



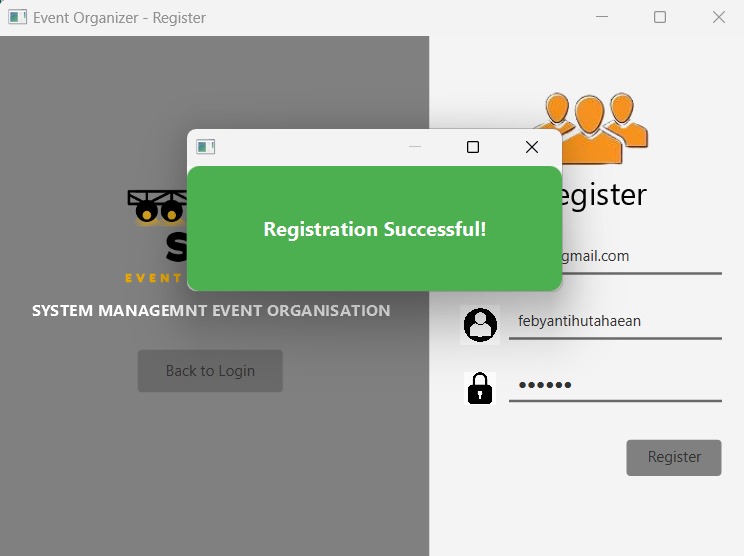
**Gambar 28 hasil implementasi register**

Jika pengguna memasukkan data yang tidak memenuhi kriteria, aplikasi akan menampilkan notifikasi seperti pada gambar berikut.



**Gambar 29 hasil implemetasi register jika tidak seusal kriteria**

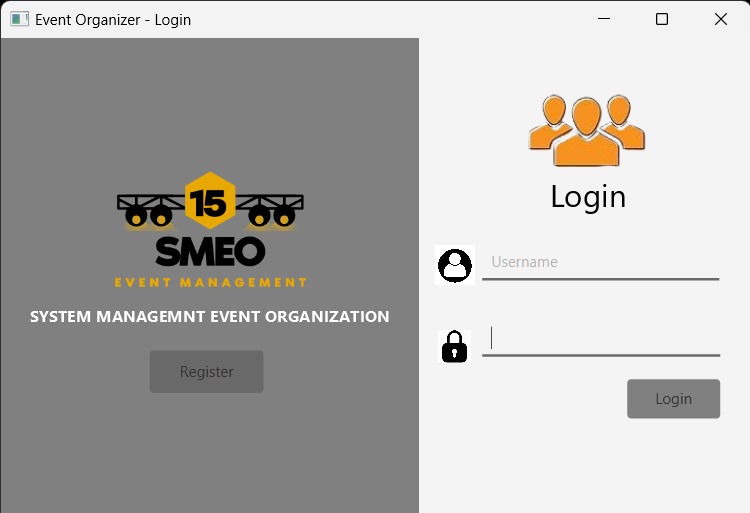
Namun, jika data yang dimasukkan benar, aplikasi akan menampilkan notifikasi seperti pada gambar berikut



**Gambar 30 hasil implementasi register jika suda berhasil buat akun**

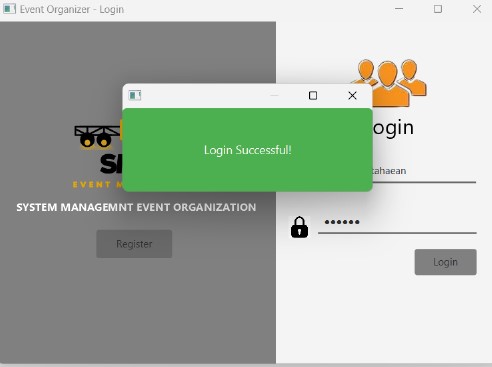
## Hasil Implementasi Login

Pada halaman ini, pengguna harus mengisi form username password, setelah itu klik button register.Akan tampil raise notice berisikan berhaisil login.



**Gambar 31 hasil implementasi login**

Jika anda sudah memasukkan username dan password dengan benar , dia akan menampilkan notifikasi seperti dibawah.

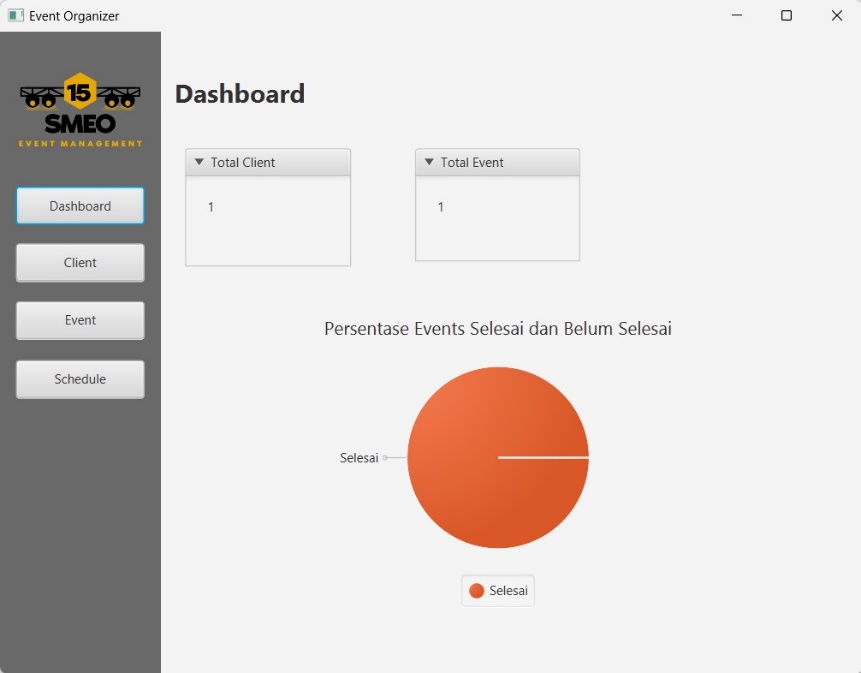


**Gambar 32 hasil implementasi login ketika berhasil masuk**

## Hasil Implementasi dashboard

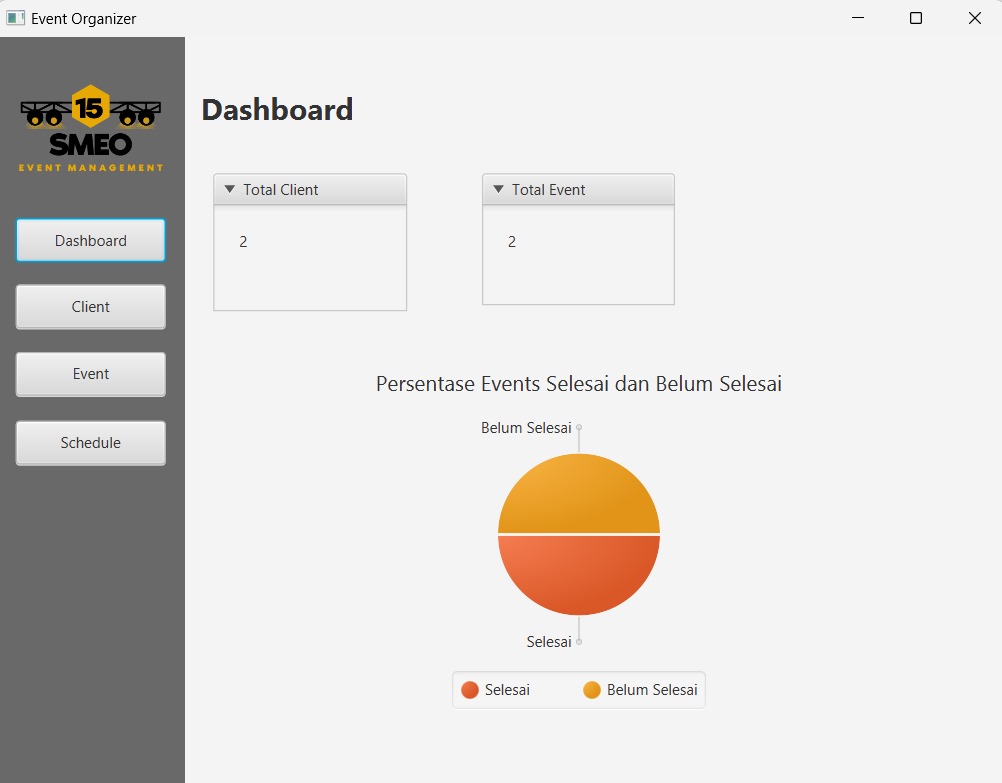
Pada halaman ini, pengguna dapat melihat jumlah client atau event , dan juga melihat presentasi event yang sudah selesai dan belum selesai. berikut ada lah hasil implementasi nya,

Berikut contoh dashboard ketika semua events selesai



**Gambar 33 hasil implementasi dashboard jika semua event selesai**

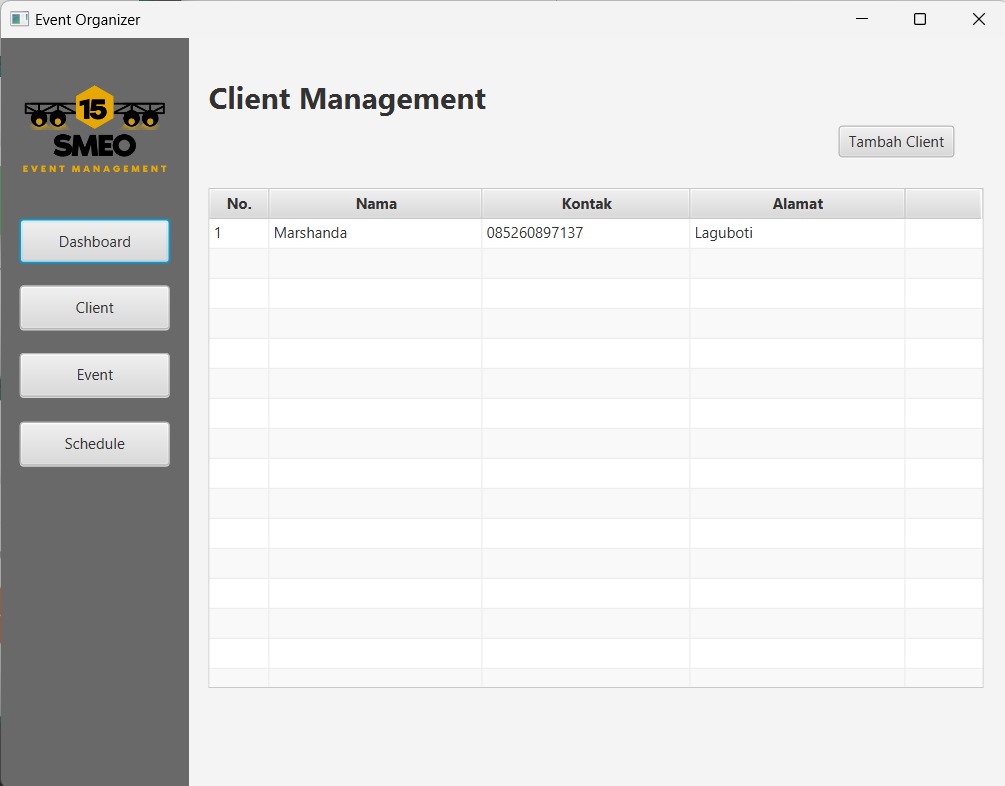
Berikut contoh dahshboard ketika ada event yang belum selesai



**Gambar 34 hasil implementasi dashboard jika sebagian event belum selesai**

## Hasil Implementasi client

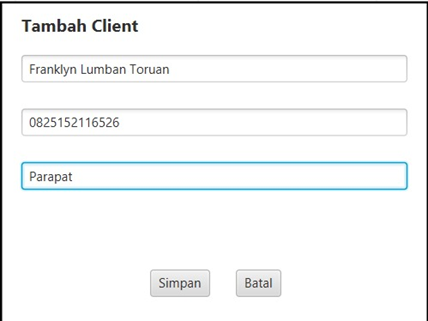
Pada halaman ini, pengguna dapat menambah client, mengedit client, dan menghapus client. Hasil implementasi dapt dilihat pada gambar dibawah



**Gambar 35 hasial implementasi client**

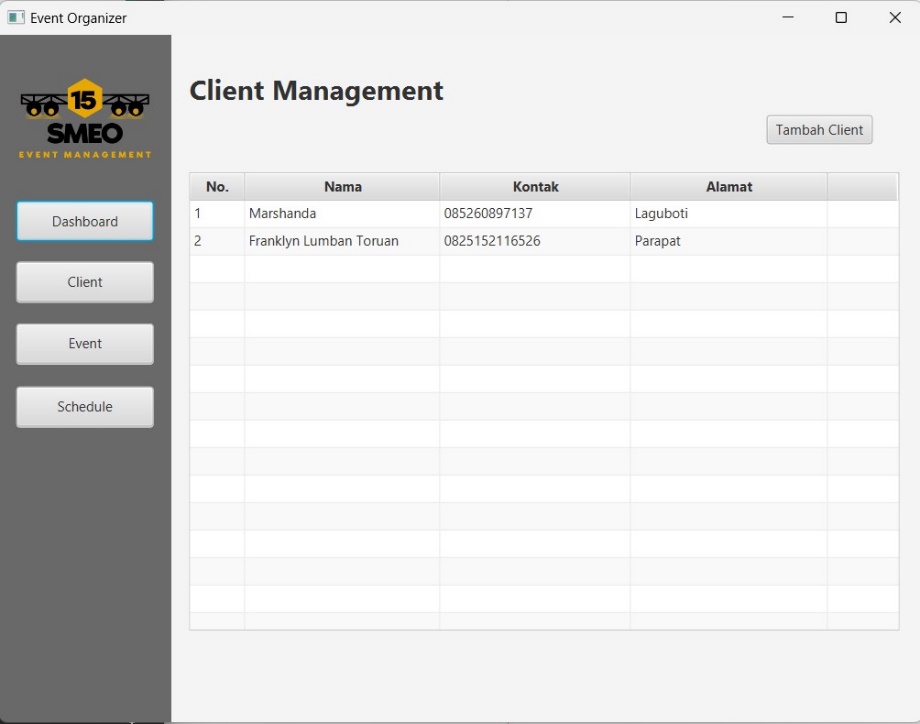
### Hasil implementasi tambah client

"Klik tombol 'Tambah Client' di atas untuk menampilkan formulir seperti di bawah ini, yang kemudian dapat diisi sesuai kebutuhan."



**Gambar 36 hasil implementasi tambah client**

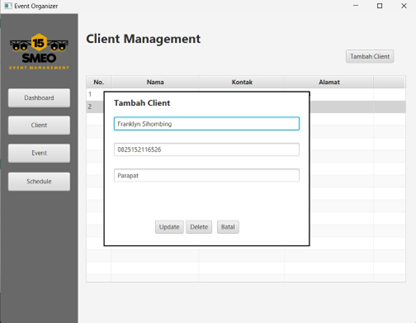
Data yang dimasukkan akan secara otomatis ditambahkan dan ditampilkan di dalam tabel



**Gambar 37 hasil implementasi jika client berhasil ditambah**

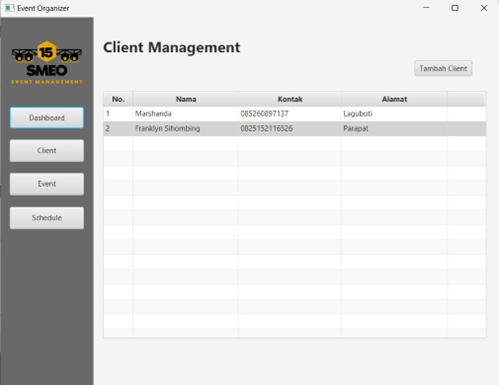
### Hasil implementasi edit client

Untuk mengedit data client, klik pada client yang ingin diubah, lalu tekan tombol 'Update' untuk memperbarui informasi sesuai kebutuhan.



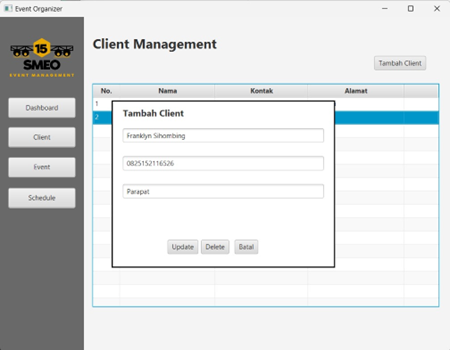
**Gambar 38 hasil implementasi edit client**

Berikut adalah hasil dari perubahan yang telah dilakukan

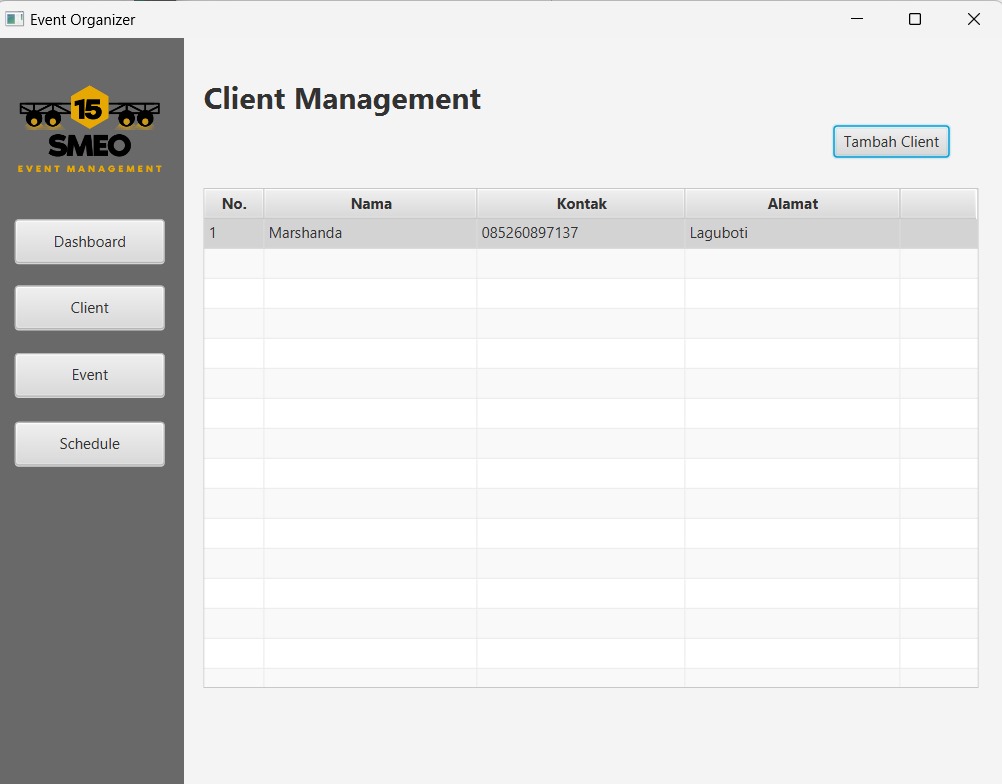
**Gambar 39 hasil implementasi jika client berhasil di edit**

### Hasil implementasi hapus client

Untuk menghapus data, klik pada data yang ingin dihapus, lalu tekan tombol 'Delete

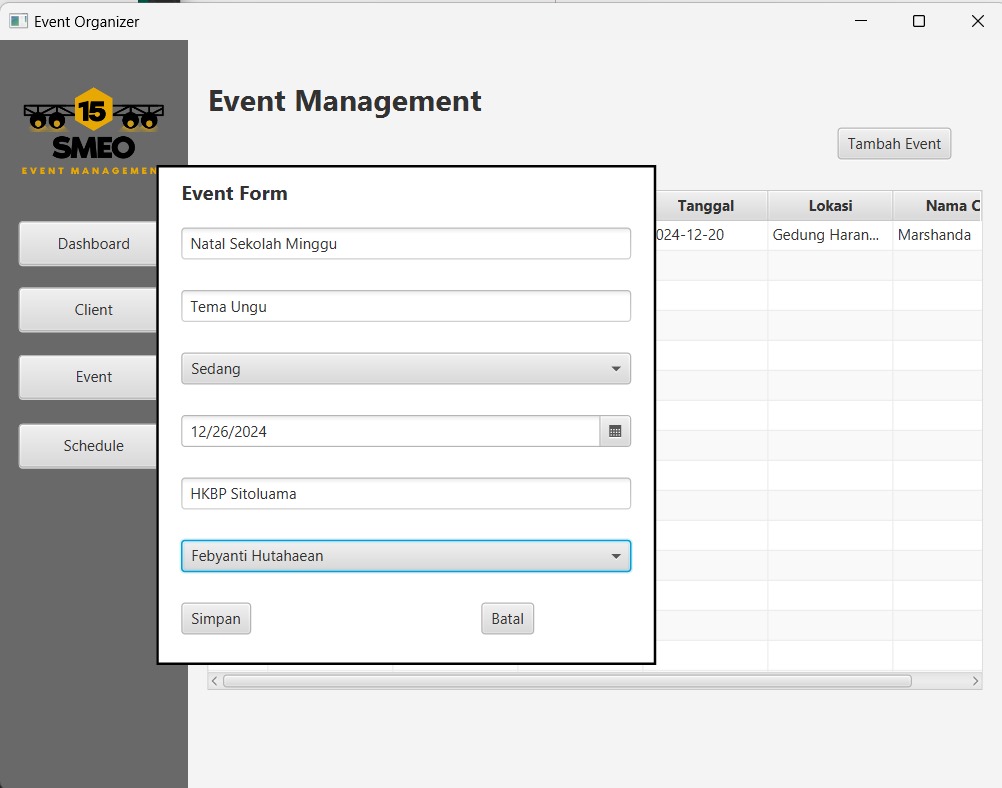
**Gambar 40 hasil implementasi hapus client**

Data tersebut akan terhapus secara otomatis

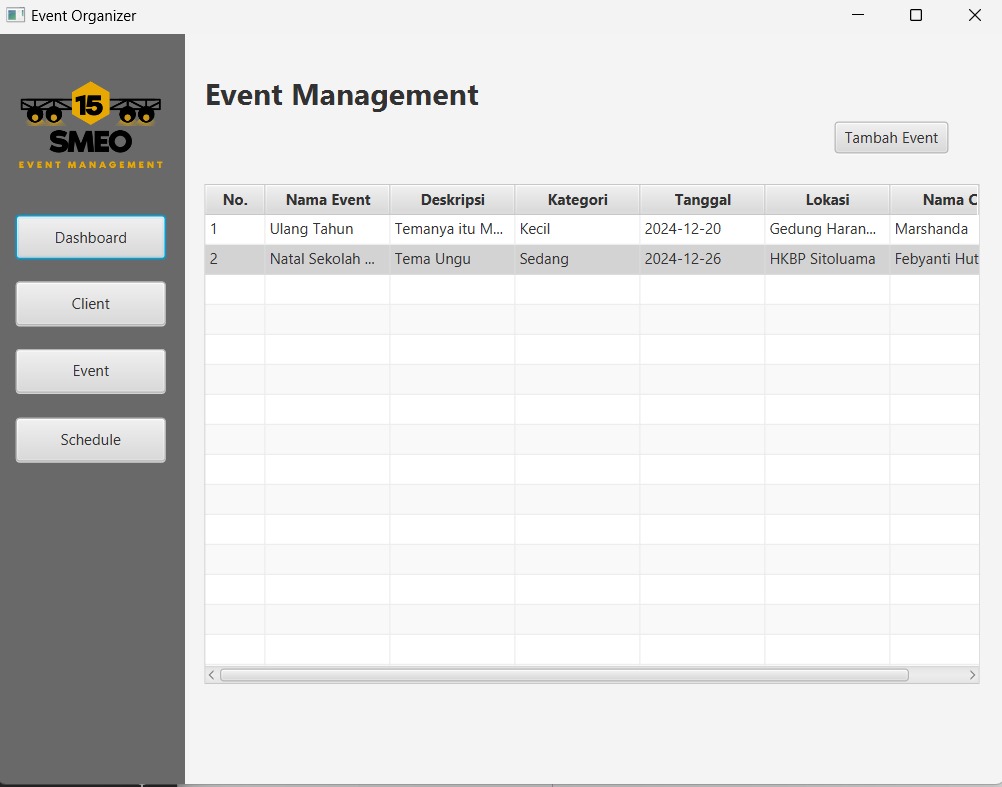
**Gambar 41 hasil implementasi hapus client**

## Hasil Implementasi Event

kita dapat menambahkan acara dengan mengisi formulir di bawah ini. Pada bagian 'Client', hanya klien yang telah ditambahkan sebelumnya melalui menu 'Tambah Client' yang dapat dipilih setlah itu ktekan simpan

**Gambar 42 implementasi tambah event**

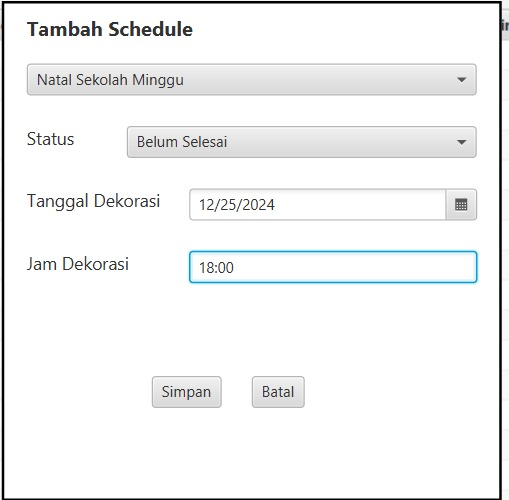
Maka event akan otomatis muncul pada tabel seperti pada gambar dibawah



**Gambar 43 hasil implementasi tambah event**

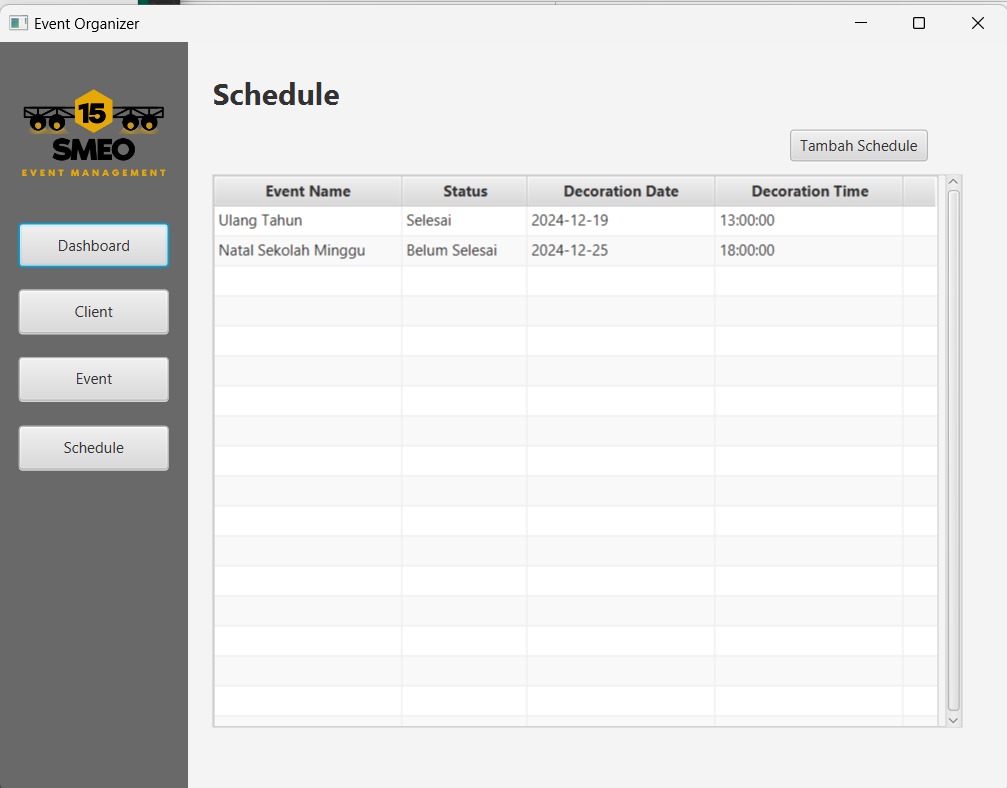
## Hasil Implementasi Schedule

Kami dapat menambahkan jadwal dengan mengisi formulir di bawah ini. Berikut adalah hasil implementasinya.



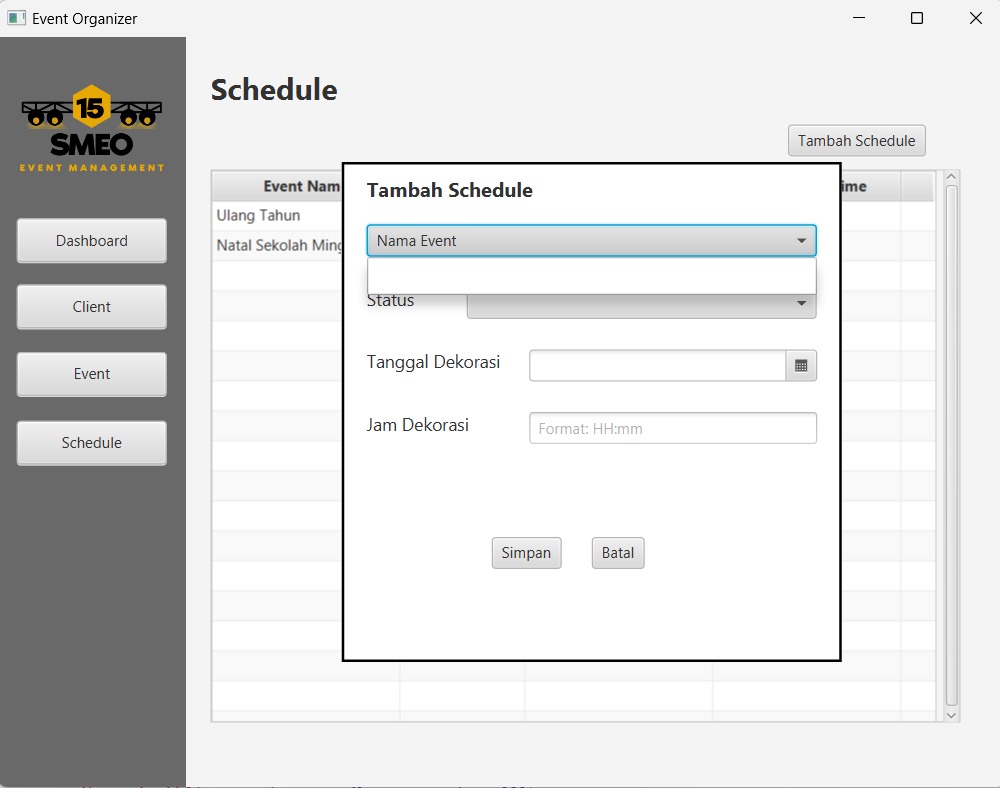
**Gambar 44 implementasi tambah schedule**

Maka akan secara otomatis akan masuk pada tabel schedule



**Gambar 45 hasil impementasi tambah schedule**

Setiap acara hanya dapat dijadwalkan sekali. Karena setiap acara sudah pernah digunakan, maka acara tersebut tidak dapat dijadwalkan kembali.seperti dibawah



**Gambar 46 implementasi jika tidak ada lagi event yang ingin ditambahakan**

# Kesimpulan Dan Saran

## Kesimpulan

Proyek "Sistem Pengelolaan Event Organizer" berhasil dikembangkan dengan menggunakan pendekatan Pemrograman Berorientasi Objek (PBO). Sistem ini dirancang untuk mempermudah pengelolaan data klien, acara, dan jadwal secara terpusat. Dengan fitur-fitur utama seperti registrasi, login, manajemen klien, manajemen acara, dan pengelolaan jadwal, sistem ini mampu meningkatkan efisiensi operasional dan akurasi dalam pengelolaan data EO. Selain itu, sistem ini menyediakan antarmuka pengguna yang intuitif dan fungsional, sehingga mempermudah pengguna dalam mengelola data dan memantau status acara secara real-time. Dari sisi pengembangan, proyek ini memberikan pengalaman langsung kepada mahasiswa dalam menerapkan konsep PBO, merancang struktur kelas, serta mengintegrasikan komponen MVC.

## Saran

1. **Pengembangan Fitur Tambahan:** Untuk meningkatkan fungsionalitas, sistem dapat dikembangkan lebih lanjut dengan menambahkan fitur notifikasi otomatis, seperti pengingat jadwal acara kepada klien atau tim EO melalui email atau pesan singkat.
2. **Integrasi Sistem:** Disarankan untuk mengintegrasikan sistem dengan platform pembayaran online untuk mendukung transaksi yang terkait dengan layanan EO.
3. **Pengujian yang Lebih Mendalam:** Lakukan pengujian secara komprehensif terhadap sistem, termasuk uji beban dan uji keamanan, untuk memastikan performa sistem tetap optimal dalam skenario penggunaan yang kompleks.
4. **Dokumentasi Pengguna:** Tambahkan panduan pengguna (user manual) yang lebih rinci untuk membantu pengguna baru memahami cara kerja sistem secara keseluruhan.
5. **Evaluasi dan Perbaikan Berkelanjutan:** Adakan evaluasi berkala dengan pengguna EO untuk mengidentifikasi area perbaikan dan meningkatkan kepuasan pengguna terhadap sistem.
6. **Penyempurnaan Antarmuka:** Meningkatkan desain antarmuka pengguna agar lebih modern dan responsif untuk pengalaman pengguna yang lebih baik, termasuk pada perangkat seluler.

# Link YT Presentasi

[LINK YT KELOMPOK 15 Sistem Pengelolaan Even Organizer](https://youtu.be/qW7mOQ2iAFM)