UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

"CARRERA DE INFORMÁTICA"

"SISTEMA DE GESTIÓN DE EVENTOS"

Estudiantes:

- Choque Quispe Juan Carlos
- Gomez Huanca Anahi Jazmin
- Valeriano Limbert
- Tarqui Blanco Yessica Nataly
- Daza Gomez Diego Alvaro
- Flores Ninachoque Jhoselyn
- Bendita Mamani Milenka Soledad

Materia: PROGRAMACION II

Docente: *Lic*. Rosalia Lopez

28 de enero de 2025

La Paz – Bolivia

1. INTRODUCCIÓN.

La organización de eventos dista mucho de ser una actividad eventual que deje algo al azar; al contrario: hasta el mínimo detalle es tenido en cuenta al momento del diseño, planificación y producción de dichas reuniones, ya sea *una convención, un congreso, una ceremonia, un festival* o cualquier otro evento empresarial o social.

Dada la gran capacidad de comunicación que poseen los eventos y los intereses financieros que pueden representar, los eventos son considerados una importante herramienta de gestión estratégica del área de marketing y de relaciones públicas de las empresas e instituciones; es por ello que es indispensable conocer cuál es el público objetivo, de qué manera se va a realizar el evento, y cuáles son las metas que se desean lograr con el mismo.

Para alcanzar el éxito en la organización de un evento empresarial, se necesita de una buena planificación y organización.

2. DESCRIPCIÓN DEL PROYECTO

El presente proyecto pretende detallar el funcionamiento de un ***"Sistema de Gestión de Eventos"*** mediante el manejo de diferentes herramientas para garantizar un buen funcionamiento además de una atención adecuada hacia los diferentes usuarios.

Así también se pretende poner en práctica los conocimientos adquiridos de la **"Programación Orientada a Objetos".**

3. OBJETIVOS.

- **OBJETIVO PRINCIPAL**

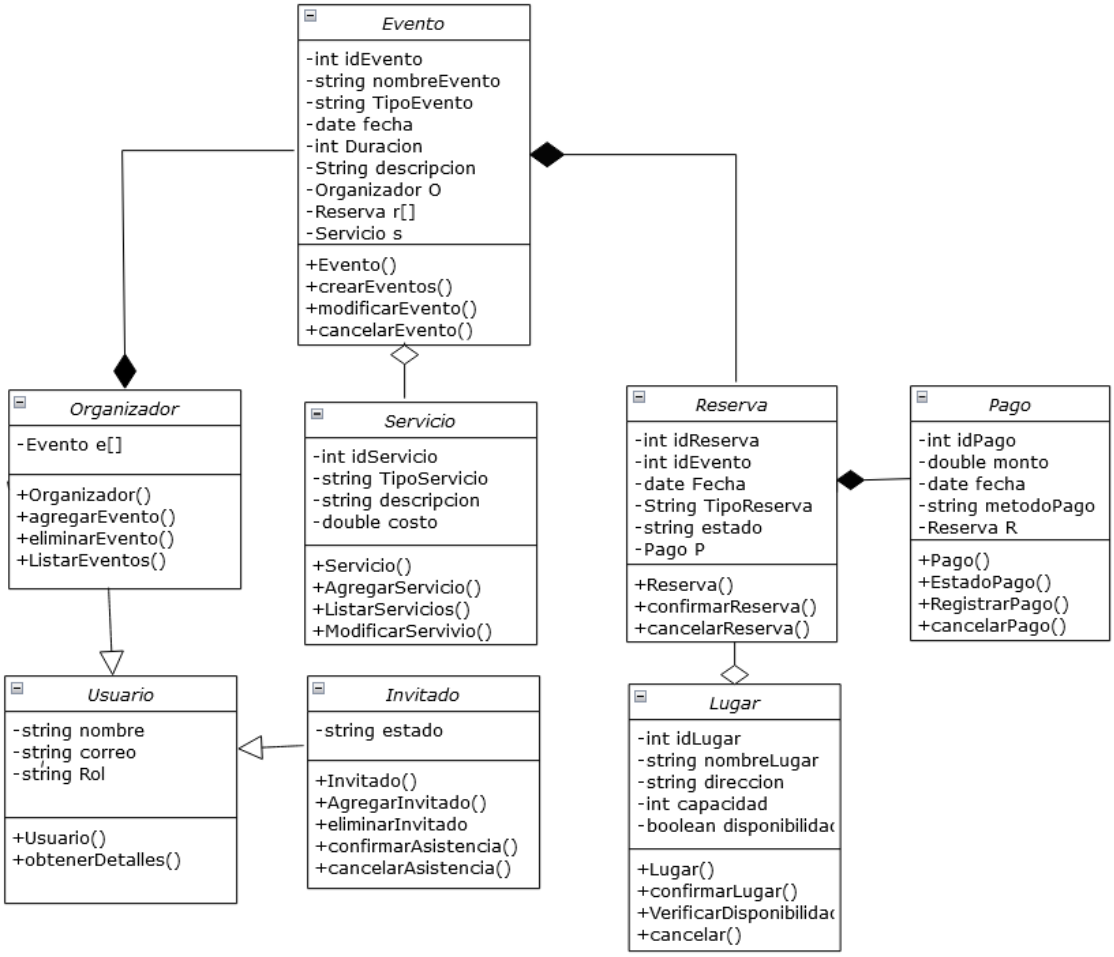
  Aplicar los conceptos fundamentales de la Programación Orientada a Objetos (POO). Diseñar un sistema modular y escalable e implementar patrones de diseño para mejorar la flexibilidad del código.

- **OBJETIVOS SECUNDARIOS**
  - ✓ Registrar eventos.
  - ✓ Ofrecer un buen servicio para los usuarios.
  - ✓ Facilitar la planificación de un evento.
  - ✓ Garantizar la correcta ejecución de un evento.

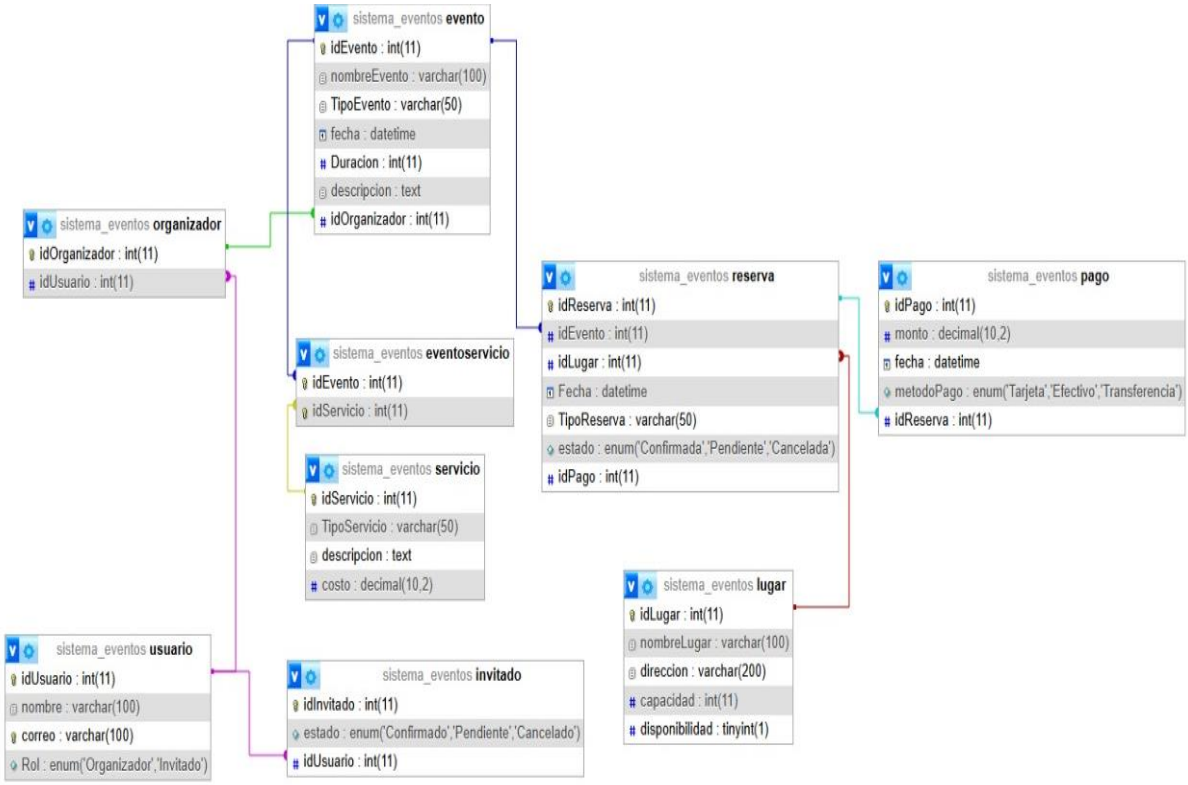
- ✓ Asegurar clientes frecuentes.

4. ANÁLISIS Y DISEÑO

- **DIAGRAMA UML.** Para la implementación de dicho proyecto se procedió a la elaboración del siguiente diagrama UML. Tomando en cuenta las clases junto a sus atributos y métodos para poder realizar una correcta implementación.



- **BASE DE DATOS**



- **PRINCIPIOS DE DISEÑO**
- ✓ **Herencia:**
- ✓ **Composición:** La clase Reserva tiene un lugar así como la clase Organizador tiene un Usuario.
- ✓ **Agregación:** Un Usuario puede estar asociado a varios eventos.

5. HERRAMIENTAS UTILIZADAS.

- **JAVA.** Es un tipo de lenguaje de programación y una plataforma, creada y comercializada por Sun Microsystems en el año 1995. Se constituye como un lenguaje orientado a objetos, su intención es permitir que los desarrolladores de aplicaciones escriban el programa una sola vez y lo ejecuten en cualquier dispositivo.
- **MySQL.** Es un sistema de administración de bases de datos relacionales. Es un software de código abierto desarrollado por Oracle. Se considera como la base de datos de código abierto más utilizada en el mundo.

- **Netbeans.** NetBeans IDE es un entorno de desarrollo integrado, gratuito y de código abierto para el desarrollo de aplicaciones en los sistemas operativos *Windows, Mac, Linux y Solaris*. El IDE simplifica el desarrollo de aplicaciones web, corporativas, de escritorio y móviles que utilizan plataformas *Java y HTML5*. El IDE también ofrece soporte para el desarrollo de aplicaciones *PHP y C/C++*.
- **POO.** La Programación Orientada a Objetos (POO) es un paradigma de programación, esto es, un modelo o un estilo de programación que proporciona unas guías acerca de cómo trabajar con él y que está basado en el concepto de clases y objetos. Este tipo de programación se emplea para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

6. IMPLEMENTACION EN JAVA.

6.1. ESTRUCTURA DEL PROYECTO.

```
public class Database {
    private static final String URL = "jdbc:mysql://localhost:3306/sistema_eventos";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    private static Connection connection;


    public static Connection getConnection() {
        if (connection == null) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                connection = DriverManager.getConnection(URL, USER, PASSWORD);
                System.out.println("Conexión exitosa a la base de datos.");
            } catch (ClassNotFoundException e) {
                System.err.println("Error: Driver no encontrado. " + e.getMessage());
            } catch (SQLException e) {
                System.err.println("Error de conexión a MySQL: " + e.getMessage());
            }
        }
        return connection;
    }

    public static void closeConnection() {
        if (connection != null) {
            try {
                connection.close();
                System.out.println("Conexión cerrada.");
            } catch (SQLException e) {
                System.err.println("Error al cerrar la conexión: " + e.getMessage());
            }
```

El código está organizado en paquetes dentro del directorio com.mycompany.gestioneventos, donde cada entidad tiene su propia clase y una clase DAO correspondiente para manejar la interacción con la base de datos.

6.2. CODIGO FUENTE.

6.3. DISEÑO DE INTERFASES.

La aplicación cuenta con una interfaz moderna desarrollada con Swing. La ventana principal (GestionEventos.java) incluye un diseño basado en pestañas (JTabbedPane), donde se presentan distintas funcionalidades:

- ✓ **Eventos:** Muestra la lista de eventos disponibles y permite ver detalles completos.
- ✓ **Lugares:** Permite verificar la disponibilidad de los lugares para eventos y realizar reservas.
- ✓ **Reservas:** Gestiona las reservas, permitiendo confirmarlas o cancelarlas.

Los datos se presentan en tablas (JTable) con un diseño personalizado para mejorar la visualización. También se implementan diálogos (JDialog) para mostrar información detallada y formularios de reserva.

CLASE EVENTO.

```
public class Evento {
    private int idEvento;
    private String nombreEvento;
    private String tipoEvento;
    private String fecha;
    private int duracion;
    private String descripcion;
    private int idOrganizador;

    public Evento(int idEvento, String nombreEvento, String tipoEvento, String fecha, int duracion, String descripcion
        this.idEvento = idEvento;
        this.nombreEvento = nombreEvento;
        this.tipoEvento = tipoEvento;
        this.fecha = fecha;
        this.duracion = duracion;
        this.descripcion = descripcion;
        this.idOrganizador = idOrganizador;
    }

    public void crearEvento() {
        EventoDAO eventoDAO = new EventoDAO();
        eventoDAO.insertarEvento(this);
    }

    public int getIdEvento() { return idEvento; }
    public String getNombreEvento() { return nombreEvento; }
    public String getTipoEvento() { return tipoEvento; }
    public String getFecha() { return fecha; }
    public int getDuracion() { return duracion; }
    public String getDescripcion() { return descripcion; }
    public int getIdOrganizador() { return idOrganizador; }
```

CLASE LUGAR.

```
public class Lugar {
    private int idLugar;
    private String nombre;
    private String direccion;
    private int capacidad;
    private boolean disponibilidad;

    public boolean isDisponibilidad() {
        return disponibilidad;
    }

    public void setDisponibilidad(boolean disponibilidad) {
        this.disponibilidad = disponibilidad;
    }


    public int getIdLugar() {
        return idLugar;
    }

    public void setIdLugar(int idLugar) {
        this.idLugar = idLugar;
    }

    public String getNombre() {
        return nombre;
    }
```

CLASE ORGANIZADOR.

```
 */
public class Organizador extends Usuario {
    private int idOrganizador;

    public Organizador() {
        this.rol = "Organizador";
    }

    @Override
    public String obtenerDetalles() {
        return String.format("Organizador ID: %d - Nombre: %s", idOrganizador, nombre);
    }

    public int getIdOrganizador() { return idOrganizador; }
    public void setIdOrganizador(int idOrganizador) { this.idOrganizador = idOrganizador; }
}
```

CLASE USUARIO.

```
public abstract class Usuario {
    protected int idUsuario;
    protected String nombre;
    protected String correo;
    protected String rol;

    public Usuario() {
    }

    public Usuario(int idUsuario, String nombre, String correo, String rol) {
        this.idUsuario = idUsuario;
        this.nombre = nombre;
        this.correo = correo;
        this.rol = rol;
    }

    public abstract String obtenerDetalles();

    public int getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getNombre() {
        return nombre;
    }
```

CLASE RESERVA

```
public class Reserva {
    private int idReserva;
    private int idEvento;
    private int idLugar;
    private LocalDateTime fecha;
    private String tipoReserva;
    private EstadoReserva estado;
    private Integer idPago;

    public Reserva() {}

    public int getIdReserva() {
        return idReserva;
    }

    public void setIdReserva(int idReserva) {
        this.idReserva = idReserva;
    }

    public int getIdEvento() {
        return idEvento;
    }

    public void setIdEvento(int idEvento) {
        this.idEvento = idEvento;
    }
```

CLASE INVITADO.

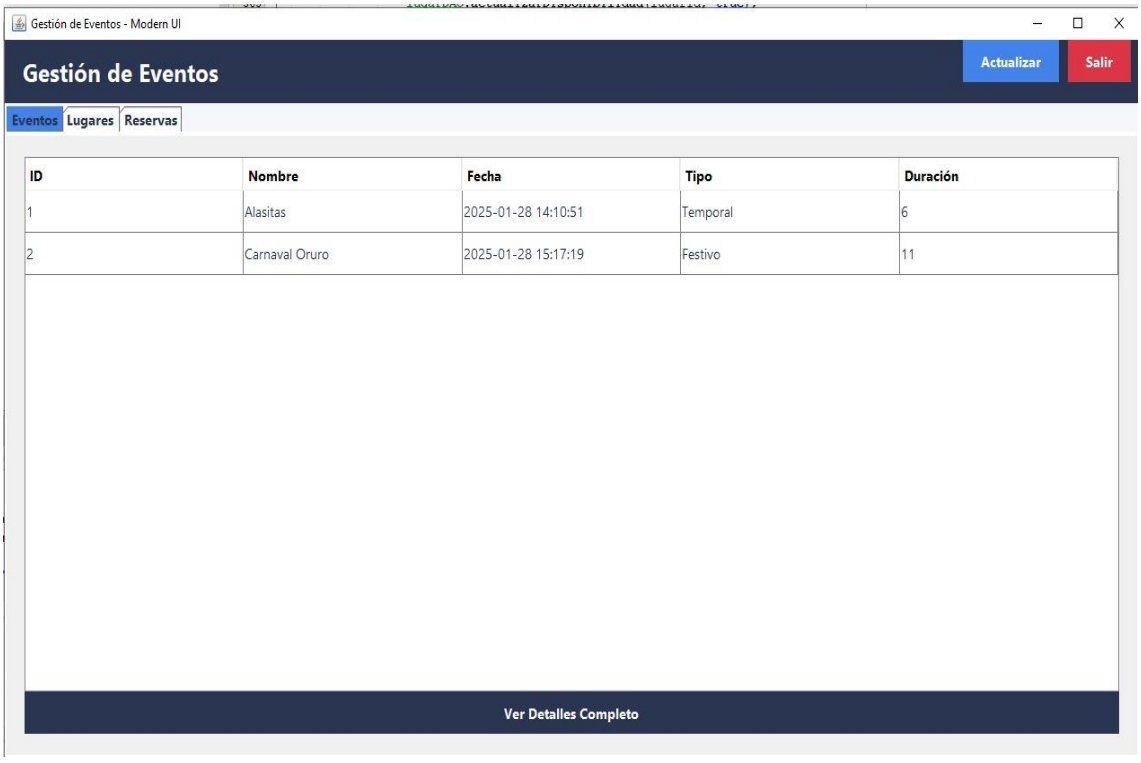
```
 */
public class Invitado extends Usuario {
    private int idInvitado;
    private Estado estado;

    public Invitado() {
        this.rol = "Invitado";
    }

    @Override
    public String obtenerDetalles() {
        return String.format("Invitado ID: %d - Estado: %s", idInvitado, estado.name());
    }

    public int getIdInvitado() { return idInvitado; }
    public void setIdInvitado(int idInvitado) { this.idInvitado = idInvitado; }
    public Estado getEstado() { return estado; }
    public void setEstado(Estado estado) { this.estado = estado; }
}
```

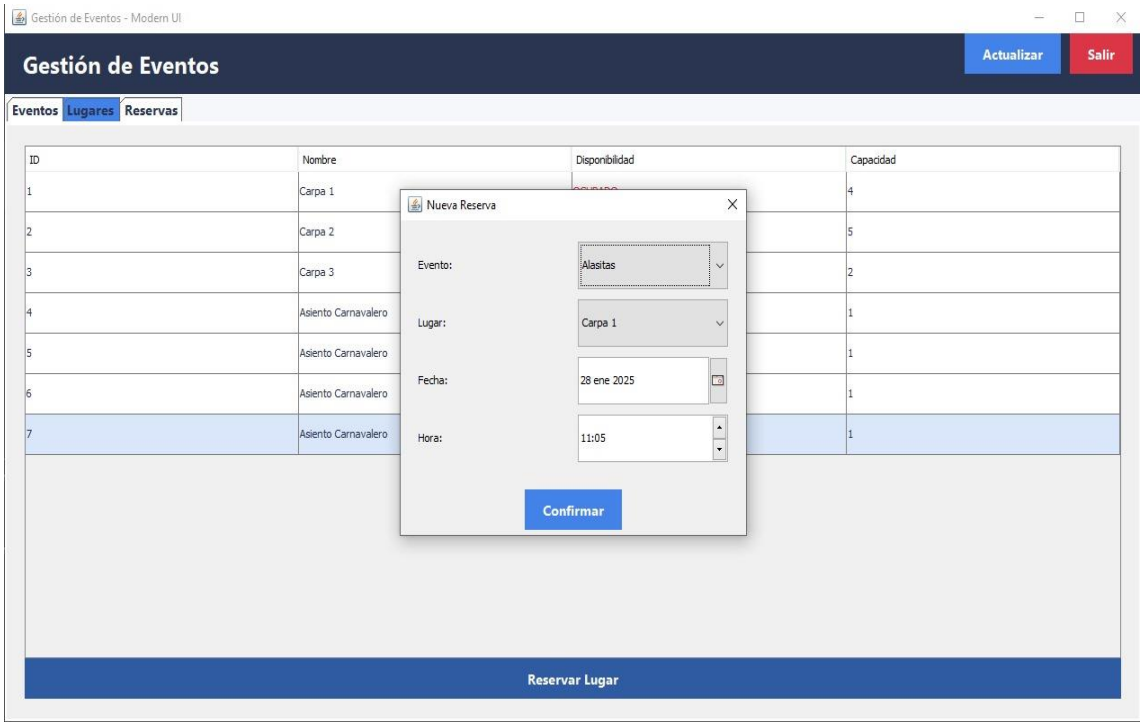
CLASE PAGO.

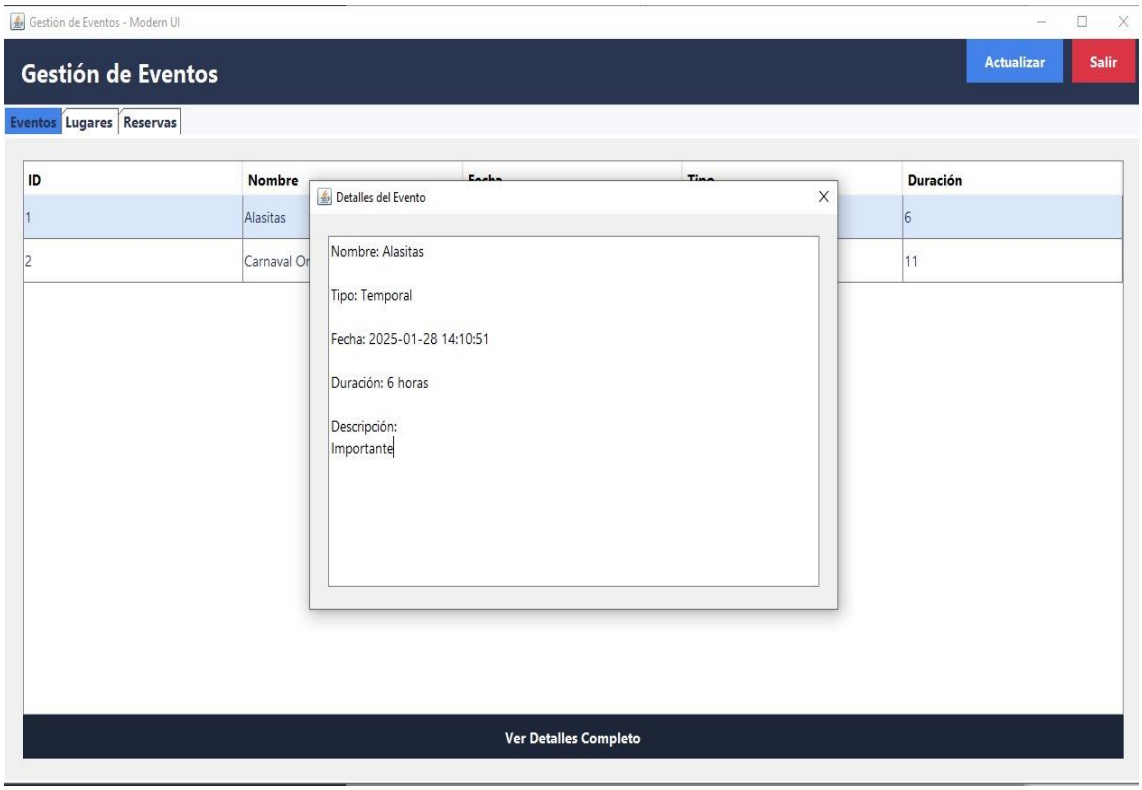
```
public class Pago {
    private int idPago;
    private BigDecimal monto;
    private LocalDateTime fecha;
    private MetodoPago metodo;
    private int idReserva;

    public Pago() {}

    public Pago(BigDecimal monto, LocalDateTime fecha, MetodoPago metodo, int idReserva) {
        this.monto = monto;
        this.fecha = fecha;
        this.metodo = metodo;
        this.idReserva = idReserva;
    }

    public int getIdPago() {
        return idPago;
    }

    public void setIdPago(int idPago) {
        this.idPago = idPago;
    }

    public BigDecimal getMonto() {
        return monto;
    }

    public void setMonto(BigDecimal monto) {
        this.monto = monto;
```

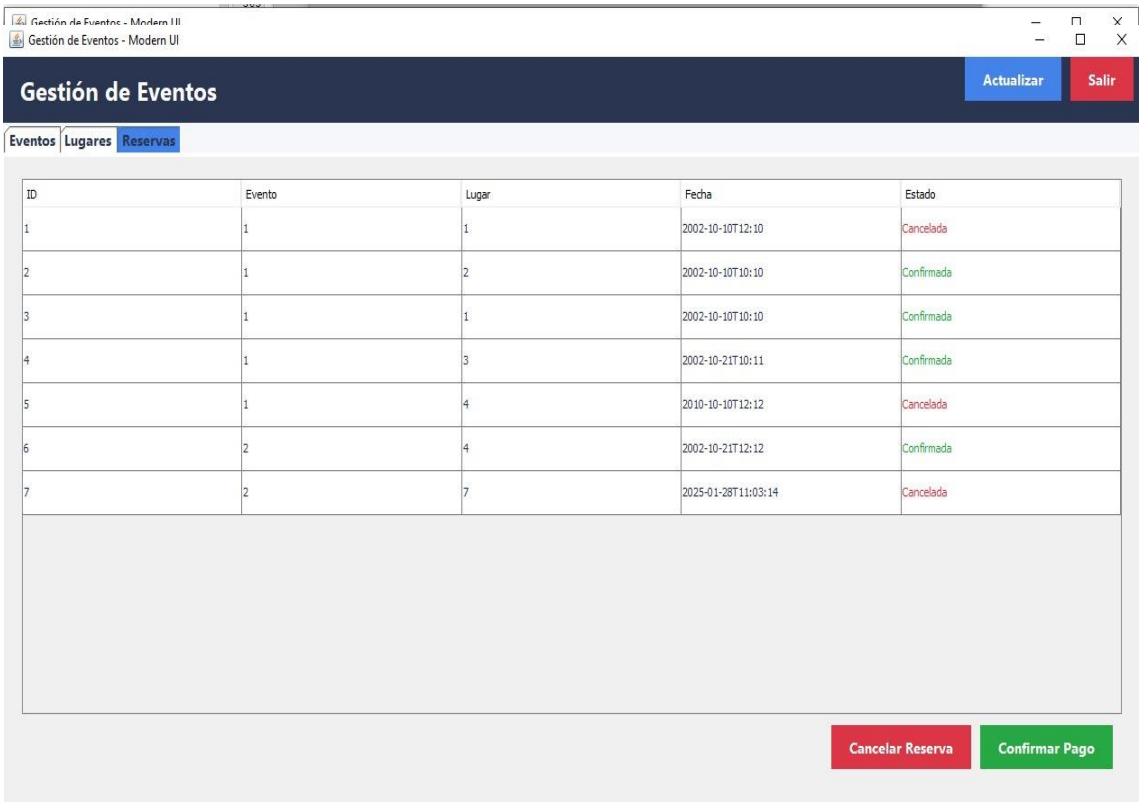
6.4. PRUEBAS DEL SISTEMA.


Invitado Form
NOMBRE:
CORREO:
ROL:
ESTADO:
GUARDAR
REGRESAR
SelectionForm ejercicios = new SelectionForm()
ejercicios.setVisible(true);


Gestión de Eventos - Modern UI
Gestión de Eventos
Actualizar
Salir
Eventos
Lugares
Reservas
ID
Nombre
Fecha
Tipo
Duración
1
Alasitas
2025-01-28 14:10:51
Temporal
6
2
Carnaval Oruro
2025-01-28 15:17:19
Festivo
11
Ver Detalles Completo


Gestión de Eventos - Modern UI
Gestión de Eventos
Actualizar
Salir
Eventos
Lugares
Reservas
ID
Nombre
Disponibilidad
Capacidad
1
Carpa 1
4
2
Carpa 2
5
3
Carpa 3
2
4
Asiento Carnavalero
1
5
Asiento Carnavalero
1
6
Asiento Carnavalero
1
7
Asiento Carnavalero
1
Nueva Reserva
Evento:
Alasitas
Lugar:
Carpa 1
Fecha:
28 ene 2025
Hora:
11:05
Confirmar
Reservar Lugar


Gestión de Eventos - Modern UI
Gestión de Eventos
Actualizar
Salir
Eventos
Lugares
Reservas
ID
Nombre
Duración
1
Alasitas
6
2
Carnaval Or
11
Detalles del Evento
Nombre: Alasitas
Tipo: Temporal
Fecha: 2025-01-28 14:10:51
Duración: 6 horas
Descripción:
Importante
Ver Detalles Completo



7. CONCLUSION.

El presente proyecto nos permite gestionar un evento de manera efectiva y dar al usuario una atención efectiva y proporcionar un servicio funcional para cumplir con las expectativas de cada uno de los usuarios así como sus gustos.