

UNIVERSIDAD MAYOR DE SAN ANDRES
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA



MATERIA: Programación II (INF-121)

DOCENTE: Lic. Rosalia Lopez Montalvo

INTEGRANTES:

Luis Miguel Condori Nina	CI:10025431
Bryan Roger Rondo Pahe	CI: 13182192
Franz Denisol Ayala Isidro	CI: 14107616
Erland Chivas Huanca	CI: 9119983
Carla Alejandra Gutierrez Garcia	CI: 13844487
Bryan Henry Alvarado Apocada	CI: 9168350

TEMA:

“Sistema de gestión de Reservas de un Hotel

Fecha: 28/01/2025

Introducción

El presente proyecto tiene como propósito el desarrollo de un Sistema de Gestión de Reservas de un Hotel utilizando los principios fundamentales de la Programación Orientada a Objetos (POO). Este sistema busca ofrecer una solución eficiente para la administración de habitaciones, clientes y reservas, integrando una interfaz gráfica amigable que facilite la interacción con el usuario.

El sistema incluye funcionalidades esenciales como el registro de clientes, consulta de disponibilidad de habitaciones, gestión de reservas (realización y cancelación), y la visualización detallada de las habitaciones reservadas y los clientes registrados. Asimismo, se ha implementado el uso de patrones de diseño y principios de arquitectura para garantizar un código escalable, modular y fácil de mantener.

1. DEFINICIÓN DEL PROYECTO

1.1 Descripción General

El Sistema de Gestión Hotelera es una aplicación desarrollada en Java con el objetivo de proporcionar una solución eficiente para la administración de reservas y operaciones de un hotel. Basado en los principios de la Programación Orientada a Objetos (POO) y empleando patrones de diseño, este sistema incluye las siguientes funcionalidades principales:

- Gestión integral del ciclo de vida de las reservas hoteleras.
- Administración de diferentes tipos de habitaciones, incluyendo Suite, Doble e Individual.
- Registro y control de clientes.
- Monitorización de la disponibilidad de habitaciones.
- Interfaz gráfica intuitiva y fácil de usar.
- Persistencia de datos mediante una base de datos MySQL.

El sistema está diseñado para optimizar la gestión de habitaciones, clientes y reservas. A través de una interfaz gráfica amigable, los usuarios podrán realizar tareas como registrar clientes, consultar la disponibilidad de habitaciones, efectuar y cancelar reservas, y acceder a información detallada sobre las habitaciones ocupadas y los clientes registrados.

1.2 Objetivos del Proyecto

Objetivo General

Desarrollar un sistema de gestión hotelera eficiente que permita automatizar y optimizar la administración de reservas, clientes y habitaciones, utilizando principios de programación moderna y tecnologías avanzadas para mejorar la experiencia del usuario y garantizar la integridad de los datos.

Objetivos Específicos

Objetivos Técnicos

- **Diseño orientado a objetos:** Crear una arquitectura robusta basada en los principios de POO, que permita una representación clara y funcional de las entidades del sistema, como habitaciones, clientes y reservas.
- **Modularidad y escalabilidad:** Desarrollar un sistema organizado en módulos que facilite su mantenimiento y expansión.
- **Integridad y persistencia de datos:** Garantizar la consistencia de la información mediante el uso de una base de datos relacional y mecanismos eficientes de acceso y almacenamiento de datos.
- **Optimización del rendimiento:** Diseñar algoritmos y estructuras de datos eficientes para operaciones críticas, como la consulta de disponibilidad de habitaciones.

- **Aplicación de patrones de diseño:** Incorporar patrones como MVC (Modelo-Vista-Controlador) para separar responsabilidades y mejorar la flexibilidad del sistema.

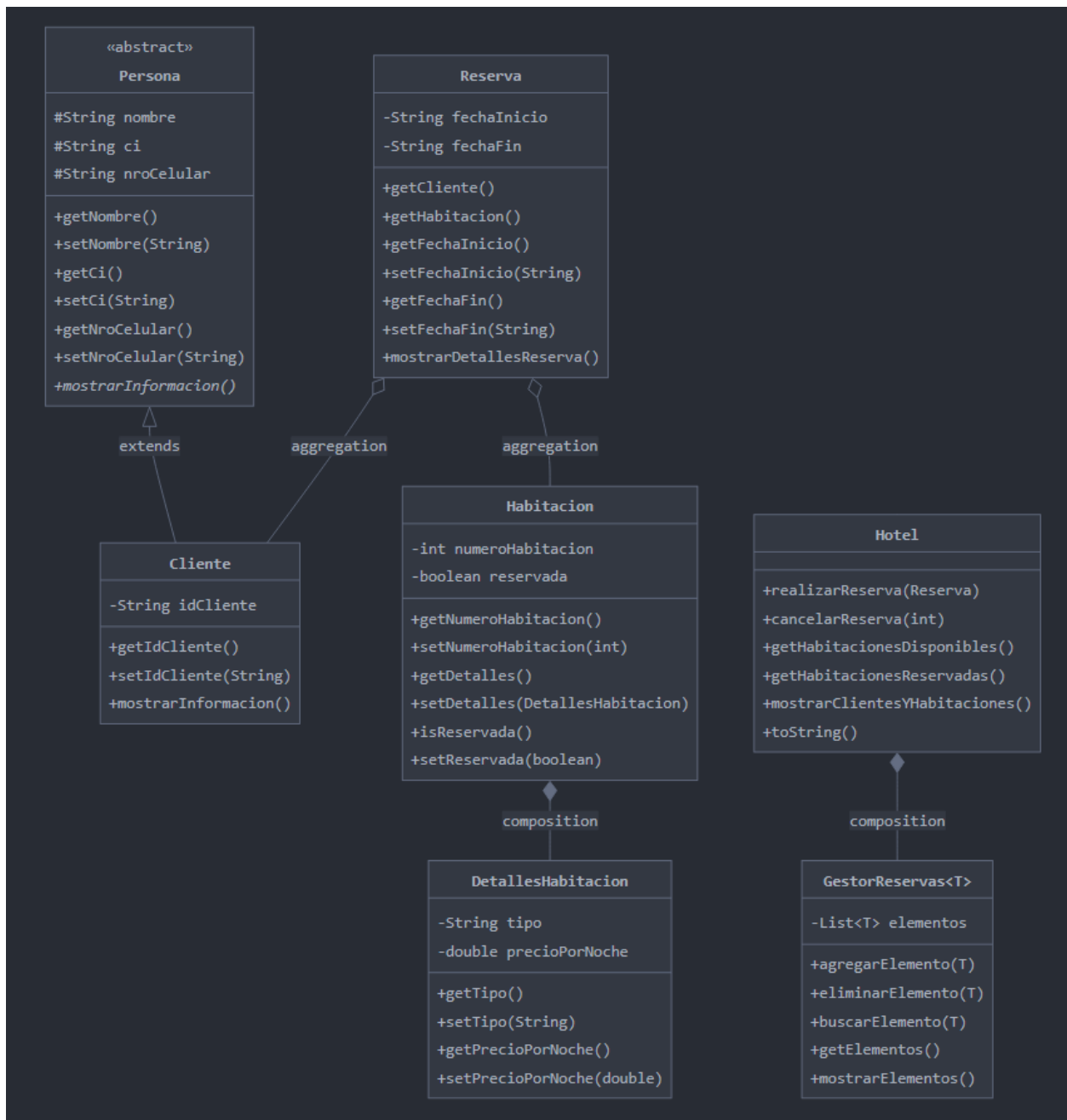
Objetivos Funcionales

- **Automatización del proceso de reservas:** Simplificar y agilizar la gestión de reservas mediante un sistema informatizado.
- **Asignación eficiente de habitaciones:** Reducir errores en la asignación y registro de habitaciones ocupadas.
- **Seguimiento de la ocupación:** Ofrecer herramientas para monitorizar la ocupación del hotel en tiempo real.
- **Experiencia de usuario mejorada:** Proporcionar una interfaz gráfica intuitiva y funcional para facilitar la interacción con el sistema.
- **Generación de reportes:** Permitir la consulta y exportación de estadísticas sobre ocupación, tipos de habitaciones más demandadas y perfil de los clientes.

Este proyecto busca satisfacer las necesidades de gestión hotelera mediante el uso de tecnologías modernas y principios de desarrollo de software sólidos, asegurando una experiencia eficiente tanto para los usuarios administrativos como para los clientes finales.

2. ANÁLISIS Y DISEÑO

2.1 Diagrama UML



2.2 Principios de Diseño

2.1 Arquitectura del Sistema

2.1.1 Patrón MVC

- **Modelo:** Clases de entidad (Cliente, Habitación, Reserva)
- **Vista:** Interfaces gráficas desarrolladas con Java Swing
- **Controlador:** Lógica de negocio y gestión de operaciones

2.1.2 Patrones de Diseño Implementados

1. Singleton

- Implementado en DatabaseConnection
- Garantiza una única instancia de conexión a la base de datos
- Optimiza el uso de recursos del sistema

2. Observer

- Utilizado para actualización en tiempo real de disponibilidad
- Notifica cambios en el estado de las habitaciones
- Mantiene sincronizada la interfaz de usuario

3. Factory Method

- Creación de diferentes tipos de habitaciones
- Facilita la extensión de nuevos tipos de habitación
- Encapsula la lógica de instanciación
-

En el análisis se identificaron las siguientes entidades principales:

- **Cliente:** Representa a los usuarios que realizan reservas.
- **Habitación:** Modela las habitaciones del hotel con atributos como tipo, precio y estado de reserva.
- **Reserva:** Relaciona a un cliente con una habitación y las fechas de inicio y fin de la estadía.
- **GestorReservas:** Una clase genérica para manejar la gestión de elementos como habitaciones y reservas.
- **Hotel:** Clase central que coordina las operaciones entre clientes, habitaciones y reservas.

El uso de herencia, composición y polimorfismo fue clave para estructurar estas clases y lograr un sistema modular y fácil de extender.

3. IMPLEMENTACIÓN EN JAVA

3.1 Estructura del Proyecto

com.mycompany.hotel/

```
|— modelo/
|   |— Persona.java
|   |— Cliente.java
|   |— Habitacion.java
|   |— DetallesHabitacion.java
|   |— Reserva.java
|— controlador/
|   |— Hotel.java
|   |— GestorReservas.java
|   |— DatabaseConnection.java
|— vista/
    |— GestorReservasHotel.java
```

3.2 Código Fuente

Clase Persona

```
package com.mycompany.hotelgestor;
abstract class Persona {
    protected String nombre;
    protected String ci; // Documento de identidad
    protected String nroCelular; // Número de celular

    public Persona(String nombre, String ci, String nroCelular) {
        this.nombre = nombre;
        this.ci = ci;
        this.nroCelular = nroCelular;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getCi() {
        return ci;
    }

    public void setCi(String ci) {
        this.ci = ci;
    }

    public String getNroCelular() {
        return nroCelular;
    }

    public void setNroCelular(String nroCelular) {
        this.nroCelular = nroCelular;
    }

    // Método abstracto que debe ser implementado por las subclases
    public abstract void mostrarInformacion();
}
```


Clase Cliente

```
package com.mycompany.hotelgestor;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class Cliente {
    private String idCliente;
    private String nombre;
    private String ci;
    private String nroCelular;
}

    public Cliente(String idCliente, String nombre, String ci, String nroCelular)
    {
        this.idCliente = idCliente;
        this.nombre = nombre;
        this.ci = ci;
        this.nroCelular = nroCelular;
    }
    public String getIdCliente() {
        return idCliente;}
    public void setIdCliente(String idCliente) {
        this.idCliente = idCliente;}
    public String getNombre() {
        return nombre;}
    public void setNombre(String nombre) {
        this.nombre = nombre;}
    public String getCi() {
        return ci;}
    public void setCi(String ci) {
        this.ci = ci;}
    public String getNroCelular() {
        return nroCelular;}
    public void setNroCelular(String nroCelular) {
        this.nroCelular = nroCelular;}
    public void guardarEnBaseDeDatos(Connection connection) {
        String query = "INSERT INTO clientes (id_cliente, nombre, ci,
nro_celular) VALUES (?, ?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setString(1, idCliente);
            ps.setString(2, nombre);
            ps.setString(3, ci);
            ps.setString(4, nroCelular);
            ps.executeUpdate();
            System.out.println("Cliente guardado en la base de datos.");
        } catch (SQLException e) {
            System.out.println("Error al guardar cliente: " + e.getMessage());
        }
    }
}
```

Clase Habitación

```
package com.mycompany.hotelgestor;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class Habitacion {
    private int numeroHabitacion;
    private String tipo;
    private double precioPorNoche;
    private boolean reservada;
    public Habitacion(int numeroHabitacion, String tipo, double precioPorNoche,
boolean reservada) {
        this.numeroHabitacion = numeroHabitacion;
        this.tipo = tipo;
        this.precioPorNoche = precioPorNoche;
        this.reservada = reservada;
    }
    public int getNumeroHabitacion() {
        return numeroHabitacion;
    }
    public void setNumeroHabitacion(int numeroHabitacion) {
        this.numeroHabitacion = numeroHabitacion;
    }
    public String getTipo() {
        return tipo;
    }
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }
    public double getPrecioPorNoche() {
        return precioPorNoche;
    }
    public void setPrecioPorNoche(double precioPorNoche) {
        this.precioPorNoche = precioPorNoche;
    }
    public boolean isReservada() {
        return reservada;
    }
    public void setReservada(boolean reservada) {
        this.reservada = reservada;
    }
    public void guardarEnBaseDeDatos(Connection connection) {
        String query = "INSERT INTO habitaciones (numero, tipo, precio,
reservada) VALUES (?, ?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setInt(1, numeroHabitacion);
            ps.setString(2, tipo);
            ps.setDouble(3, precioPorNoche);
            ps.setBoolean(4, reservada);
            ps.executeUpdate();
            System.out.println("Habitación guardada en la base de datos.");
        }
    }
}
```

```

    } catch (SQLException e) {
        System.out.println("Error al guardar habitación: " + e.getMessage());
    }
}

public static List<Habitacion> obtenerDisponibles(Connection connection) {
    String query = "SELECT numero, tipo, precio, reservada FROM habitaciones
WHERE reservada = false";
    List<Habitacion> disponibles = new ArrayList<>();
    try (PreparedStatement ps = connection.prepareStatement(query);
        ResultSet rs = ps.executeQuery()) {

        while (rs.next()) {
            int numero = rs.getInt("numero");
            String tipo = rs.getString("tipo");
            double precio = rs.getDouble("precio");
            boolean reservada = rs.getBoolean("reservada");
            // Crear una instancia de Habitacion y agregarla a La Lista
            Habitacion habitacion = new Habitacion(numero, tipo, precio,
reservada);
            disponibles.add(habitacion);
        }

    } catch (SQLException e) {
        System.out.println("Error al obtener habitaciones disponibles: " +
e.getMessage());
    }
    return disponibles;
}

public static List<Habitacion> obtenerReservadas(Connection connection) {
    String query = "SELECT numero, tipo, precio, reservada FROM habitaciones
WHERE reservada = true";
    List<Habitacion> reservadas = new ArrayList<>();

    try (PreparedStatement ps = connection.prepareStatement(query);
        ResultSet rs = ps.executeQuery()) {

        while (rs.next()) {
            int numero = rs.getInt("numero");
            String tipo = rs.getString("tipo");
            double precio = rs.getDouble("precio");
            boolean reservada = rs.getBoolean("reservada");

            // Crear una instancia de Habitacion y agregarla a La Lista
            Habitacion habitacion = new Habitacion(numero, tipo, precio,
reservada);
            reservadas.add(habitacion);
        }

    } catch (SQLException e) {
        System.out.println("Error al obtener habitaciones reservadas: " +
e.getMessage());
    }
    return reservadas;}}

```

Clase DetallesHabitacion

```
package com.mycompany.hotelgestor;
// Clase que contiene detalles sobre una habitación
class DetallesHabitacion {
    private String tipo;
    private double precioPorNoche;

    public DetallesHabitacion(String tipo, double precioPorNoche) {
        this.tipo = tipo;
        this.precioPorNoche = precioPorNoche;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public double getPrecioPorNoche() {
        return precioPorNoche;
    }

    public void setPrecioPorNoche(double precioPorNoche) {
        this.precioPorNoche = precioPorNoche;
    }
}
```

Clase Reserva

```
package com.mycompany.hotelgestor;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class Reserva {
    private String idReserva;
    private String idCliente;
    private int idHabitacion;
    private String fechaInicio;
    private String fechaFin;
    public Reserva(String idReserva, String idCliente, int idHabitacion, String
fechaInicio, String fechaFin) {
        this.idReserva = idReserva;
        this.idCliente = idCliente;
        this.idHabitacion = idHabitacion;
    }
}
```

```

        this.fechaInicio = fechaInicio;
        this.fechaFin = fechaFin;
    }
    public String getIdReserva() {
        return idReserva;
    }
    public void setIdReserva(String idReserva) {
        this.idReserva = idReserva;
    }
    public String getIdCliente() {
        return idCliente;
    }
    public void setIdCliente(String idCliente) {
        this.idCliente = idCliente;
    }
    public int getIdHabitacion() {
        return idHabitacion;
    }
    public void setIdHabitacion(int idHabitacion) {
        this.idHabitacion = idHabitacion;
    }
    public String getFechaInicio() {
        return fechaInicio;
    }
    public void setFechaInicio(String fechaInicio) {
        this.fechaInicio = fechaInicio;
    }
    public String getFechaFin() {
        return fechaFin;
    }
    public void setFechaFin(String fechaFin) {
        this.fechaFin = fechaFin;
    }
    public void guardarEnBaseDeDatos(Connection connection) {
        String query = "INSERT INTO reservas (id_reserva, id_cliente,
id_habitacion, fecha_inicio, fecha_fin) VALUES (?, ?, ?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setString(1, idReserva);
            ps.setString(2, idCliente);
            ps.setInt(3, idHabitacion);
            ps.setString(4, fechaInicio);
            ps.setString(5, fechaFin);
            ps.executeUpdate();
            System.out.println("Reserva guardada en la base de datos.");
        } catch (SQLException e) {
            System.out.println("Error al guardar reserva: " + e.getMessage());
        }
    }
    public static List<Reserva> obtenerReservas(Connection connection) {
        String query = "SELECT id_reserva, id_cliente, id_habitacion,
fecha_inicio, fecha_fin FROM reservas";
        List<Reserva> reservas = new ArrayList<>();

        try (PreparedStatement ps = connection.prepareStatement(query);
            ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                String idReserva = rs.getString("id_reserva");
                String idCliente = rs.getString("id_cliente");
                int idHabitacion = rs.getInt("id_habitacion");
                String fechaInicio = rs.getString("fecha_inicio");
                String fechaFin = rs.getString("fecha_fin");
            }
        }
    }

```

```

        // Crear una nueva instancia de Reserva y agregarla a La Lista
        Reserva reserva = new Reserva(idReserva, idCliente, idHabitacion,
fechaInicio, fechaFin);
        reservas.add(reserva);
    } catch (SQLException e) {
        System.out.println("Error al obtener reservas: " + e.getMessage());
    }
    return reservas;
}

public static void eliminarReserva(Connection connection, String idReserva) {
    String query = "DELETE FROM reservas WHERE id_reserva = ?";
    try (PreparedStatement ps = connection.prepareStatement(query)) {
        ps.setString(1, idReserva);
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Reserva con ID " + idReserva + " eliminada
exitosamente.");
        } else {
            System.out.println("No se encontró ninguna reserva con el ID " +
idReserva + ".");
        }
    } catch (SQLException e) {
        System.out.println("Error al eliminar reserva: " + e.getMessage());
    }
}
}
}
}

```

Clase Controlador: Hotel

```

package com.mycompany.hotelgestor;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class Hotel {
    private int idHotel;
    private String nombre;
    private String direccion;
    public Hotel(int idHotel, String nombre, String direccion) {
        this.idHotel = idHotel;
        this.nombre = nombre;
        this.direccion = direccion;
    }
    public int getIdHotel() {
        return idHotel;
    }
    public void setIdHotel(int idHotel) {
        this.idHotel = idHotel;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getDireccion() {
        return direccion;
    }
    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }
    public void guardarEnBaseDeDatos(Connection connection) {

```

```

        String query = "INSERT INTO hoteles (id_hotel, nombre, direccion) VALUES
        (?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setInt(1, idHotel);
            ps.setString(2, nombre);
            ps.setString(3, direccion);
            ps.executeUpdate();
            System.out.println("Hotel guardado en la base de datos.");
        } catch (SQLException e) {
            System.out.println("Error al guardar hotel: " + e.getMessage());
        }
    }

    public void actualizarEnBaseDeDatos(Connection connection) {
        String query = "UPDATE hoteles SET nombre = ?, direccion = ? WHERE
        id_hotel = ?";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setString(1, nombre);
            ps.setString(2, direccion);
            ps.setInt(3, idHotel);
            ps.executeUpdate();
            System.out.println("Hotel actualizado en la base de datos.");
        } catch (SQLException e) {
            System.out.println("Error al actualizar hotel: " + e.getMessage());
        }
    }

    public void eliminarEnBaseDeDatos(Connection connection) {
        String query = "DELETE FROM hoteles WHERE id_hotel = ?";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setInt(1, idHotel);
            ps.executeUpdate();
            System.out.println("Hotel eliminado de la base de datos.");
        } catch (SQLException e) {
            System.out.println("Error al eliminar hotel: " + e.getMessage());
        }
    }
}

```

Clase Controlador GestorReservas

```
package com.mycompany.hotelgestor;
import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;
import java.util.ArrayList;
import java.util.List;
public class GestorReservas<T> {
    private List<T> elementos;

    public GestorReservas() {
        this.elementos = new ArrayList<T>();
    }

    public void agregarElemento(T elemento) {
        elementos.add(elemento);
    }

    public List<T> getElementos() {
        return elementos;
    }

    // BUSCA EN EL VECTOR DE HABITACIONES Y ELIMINA ELEMENTO
    public boolean eliminarElemento(T elemento) {
        return elementos.remove(elemento);
    }

    //BUSCA CASILLA DEL ELEMENTO
    public T buscarElemento(T criterio) {
        for (int i = 0; i < elementos.size(); i++) {
            T elemento = elementos.get(i);
            if (elemento.equals(criterio)) {
                return elemento;
            }
        }
        return null;
    }

    public void mostrarElementos() {
        for (int i = 0; i < elementos.size(); i++) {
            T elemento = elementos.get(i);
            System.out.println(elemento.toString());
        }
    }
}
```


Clase Contralador DatabaseConnection

```
package com.mycompany.hotelgestor;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    // Datos de conexión
    private static final String URL =
"jdbc:mariadb://localhost:3306/hotel_reserva";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    // Método para obtener una conexión
    public static Connection getConnection() {
        Connection connection = null;
        try {
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Conexión exitosa a la base de datos.");
        } catch (SQLException e) {
            System.out.println("Error al conectar a la base de datos: " +
e.getMessage());
        }
        return connection;
    }

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

Clase Vista GestorReservasHotel

```
package com.mycompany.hotelgestor;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

public class GestorReservasHotel extends JFrame {
    private JTextArea outputArea;

    public GestorReservasHotel() {
        setTitle("Gestor de Reservas de Hotel");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        outputArea = new JTextArea();
        outputArea.setEditable(false);
        add(new JScrollPane(outputArea), BorderLayout.CENTER);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(7, 1));

        JButton btnRegistrarCliente = new JButton("Registrar cliente");
        btnRegistrarCliente.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                registrarCliente();
            }
        });
        panel.add(btnRegistrarCliente);

        JButton btnMostrarDisponibles = new JButton("Mostrar habitaciones disponibles");
        btnMostrarDisponibles.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                mostrarHabitacionesDisponibles();
            }
        });
        panel.add(btnMostrarDisponibles);

        JButton btnMostrarReservadas = new JButton("Mostrar habitaciones reservadas");
        btnMostrarReservadas.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                mostrarHabitacionesReservadas();
            }
        });
        panel.add(btnMostrarReservadas);
    }
}
```

```

    }
});
panel.add(btnMostrarReservadas);

JButton btnReservarHabitacion = new JButton("Reservar habitación");
btnReservarHabitacion.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        reservarHabitacion();
    }
});
panel.add(btnReservarHabitacion);

JButton btnCancelarReserva = new JButton("Cancelar reserva");
btnCancelarReserva.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        cancelarReserva();
    }
});
panel.add(btnCancelarReserva);

JButton btnMostrarClientes = new JButton("Mostrar clientes y habitaciones reservadas");
btnMostrarClientes.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        mostrarClientesYHabitaciones();
    }
});
panel.add(btnMostrarClientes);

JButton btnSalir = new JButton("Salir");
btnSalir.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
panel.add(btnSalir);

add(panel, BorderLayout.WEST);
}

private void registrarCliente() {
    String nombre = JOptionPane.showInputDialog(this, "Ingrese su nombre:");
    String ci = JOptionPane.showInputDialog(this, "Ingrese su CI:");
    String nroCelular = JOptionPane.showInputDialog(this, "Ingrese su número de celular:");

    Cliente cliente = new Cliente(null, nombre, ci, nroCelular); // ID se genera en la base de datos

```

```

        try (Connection connection = DBConnection.getConnection()) {
            cliente.guardarEnBaseDeDatos(connection);
            outputArea.append("Cliente registrado exitosamente.\n");
        } catch (Exception e) {
            outputArea.append("Error al registrar cliente: " + e.getMessage() +
"\n");
        }
    }

    private void mostrarHabitacionesDisponibles() {
        try (Connection connection = DBConnection.getConnection()) {
            List<Habitacion> disponibles =
Habitacion.obtenerDisponibles(connection);
            outputArea.append("\nHabitaciones disponibles:\n");
            for (Habitacion h : disponibles) {
                outputArea.append("Habitación " + h.getNumeroHabitacion() + " - "
+ h.getTipo() + "\n");
            }
        } catch (SQLException e) {
            outputArea.append("Error al obtener habitaciones disponibles: " +
e.getMessage() + "\n");
        }
    }

    private void mostrarHabitacionesReservadas() {
        try (Connection connection = DBConnection.getConnection()) {
            List<Habitacion> reservadas =
Habitacion.obtenerReservadas(connection);
            outputArea.append("\nHabitaciones reservadas:\n");
            for (Habitacion h : reservadas) {
                outputArea.append("Habitación " + h.getNumeroHabitacion() + " - "
+ h.getTipo() + "\n");
            }
        } catch (SQLException e) {
            outputArea.append("Error al obtener habitaciones reservadas: " +
e.getMessage() + "\n");
        }
    }

    private void reservarHabitacion() {
        try (Connection connection = DBConnection.getConnection()) {
            String numeroStr = JOptionPane.showInputDialog(this, "Seleccione el
número de habitación:");
            int numero = Integer.parseInt(numeroStr);

            String inicio = JOptionPane.showInputDialog(this, "Ingrese fecha
inicio (YYYY-MM-DD):");
            String fin = JOptionPane.showInputDialog(this, "Ingrese fecha fin
(YYYY-MM-DD):");

            String clienteId = JOptionPane.showInputDialog(this, "Ingrese el ID
del cliente:");

```

```

        Reserva reserva = new Reserva(null, clienteId, numero, inicio, fin);
        reserva.guardarEnBaseDeDatos(connection);

        outputArea.append("Reserva realizada exitosamente.\n");
    } catch (SQLException e) {
        outputArea.append("Error al realizar la reserva: " + e.getMessage() +
"\n");
    } catch (NumberFormatException e) {
        outputArea.append("Número de habitación inválido.\n");
    }
}

private void cancelarReserva() {
    try (Connection connection = DBConnection.getConnection()) {
        String reservaId = JOptionPane.showInputDialog(this, "Ingrese el ID
de la reserva a cancelar:");
        Reserva.eliminarReserva(connection, reservaId);
        outputArea.append("Reserva cancelada exitosamente.\n");
    } catch (SQLException e) {
        outputArea.append("Error al cancelar la reserva: " + e.getMessage() +
"\n");
    }
}

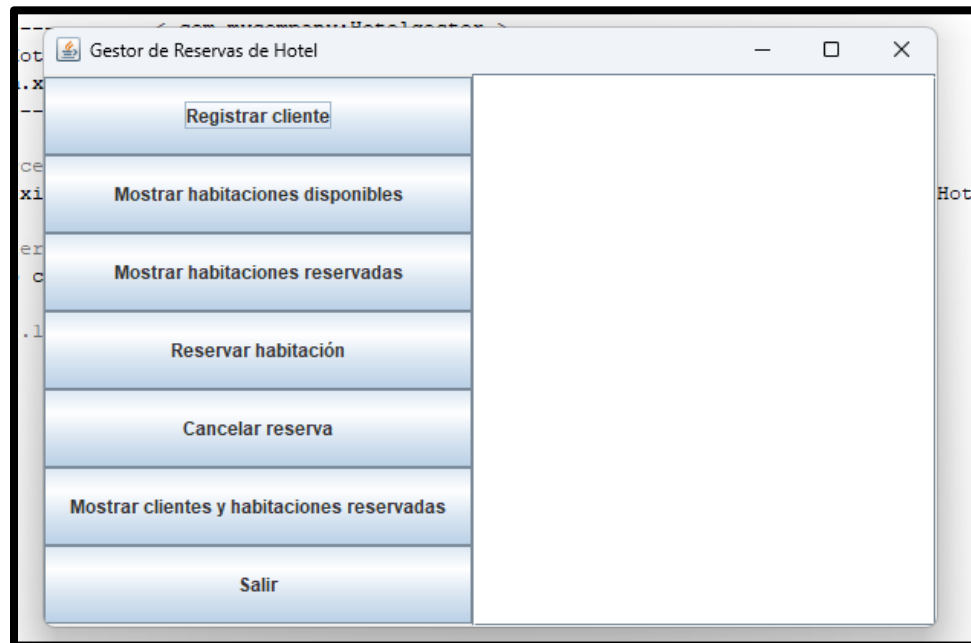
private void mostrarClientesYHabitaciones() {
    try (Connection connection = DBConnection.getConnection()) {
        List<Reserva> reservas = Reserva.obtenerReservas(connection);
        outputArea.append("\nClientes y sus habitaciones reservadas:\n");
        for (Reserva r : reservas) {
            outputArea.append("Cliente " + r.getIdCliente() + " - Habitación
" + r.getIdHabitacion() + "\n");
        }
    } catch (SQLException e) {
        outputArea.append("Error al mostrar clientes y habitaciones: " +
e.getMessage() + "\n");
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new GestorReservasHotel().setVisible(true);
        }
    });
}
}

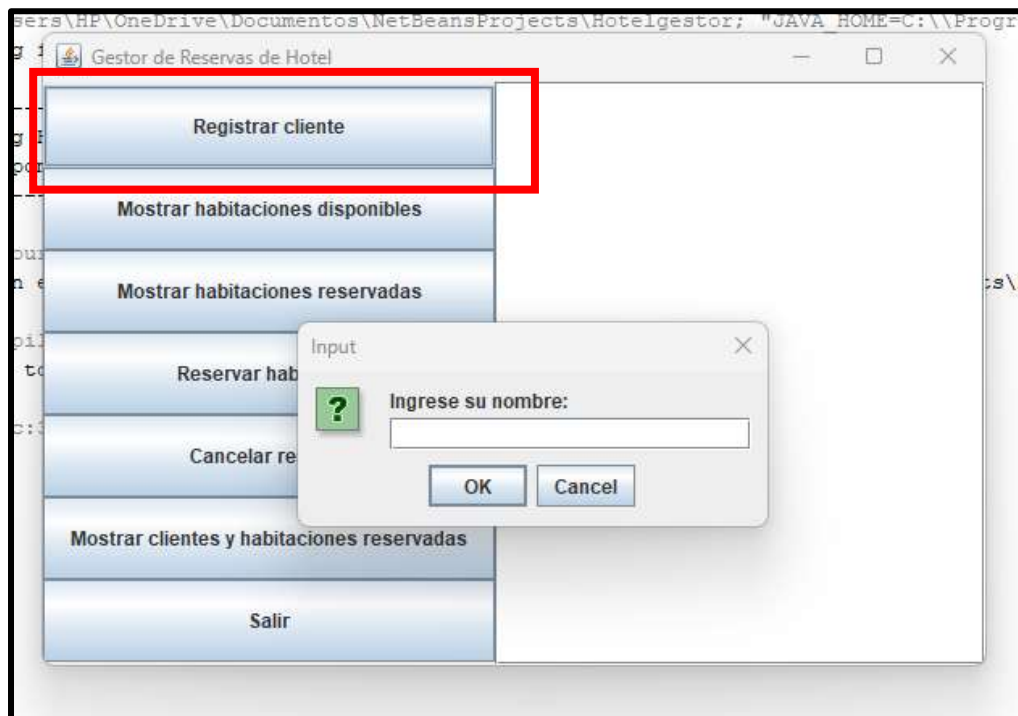
```

3.3 Diseño de Interfaces

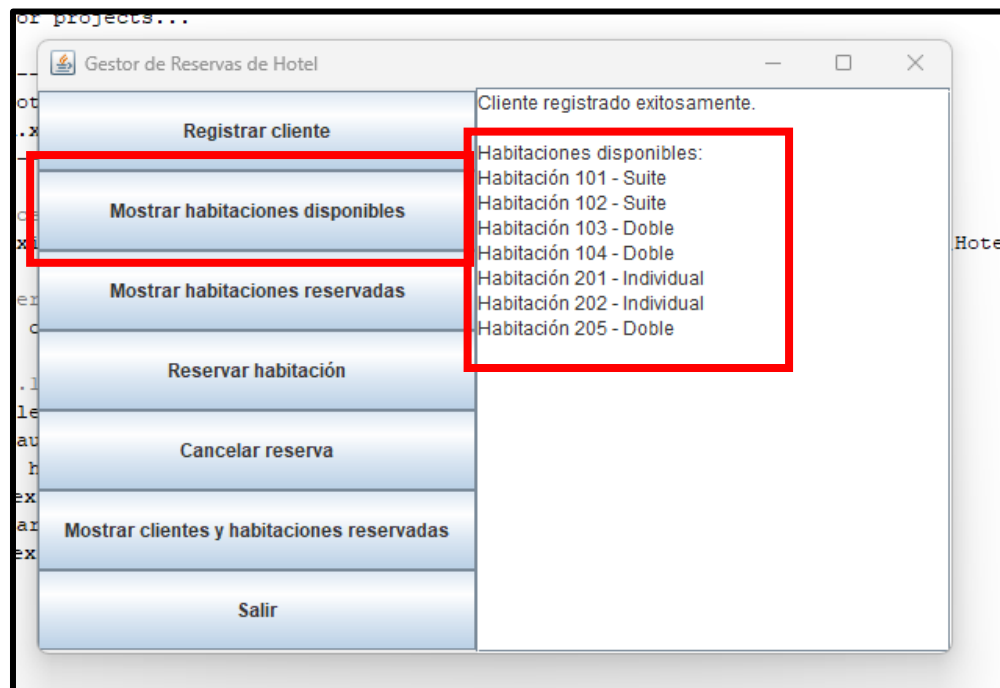
Pantalla principal de la interfaz grafica



Opción de Registrar clientes: nos solicitara datos como el nombre, CI y el numero de celular del cliente



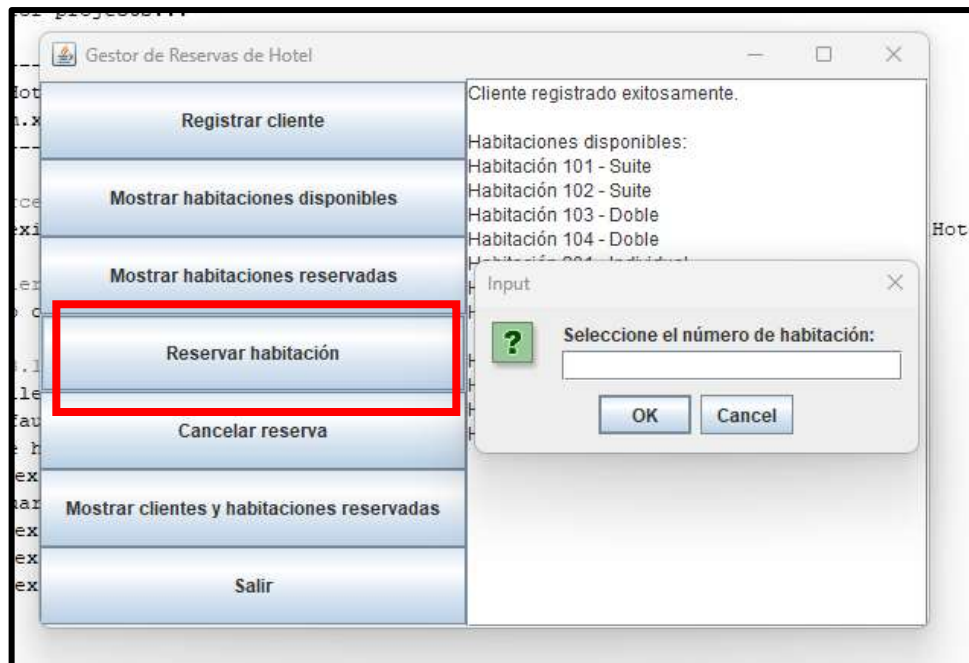
Mostrar habitaciones disponibles nos da la opción de verificar cuales son las habitaciones que tenemos disponibles al momento



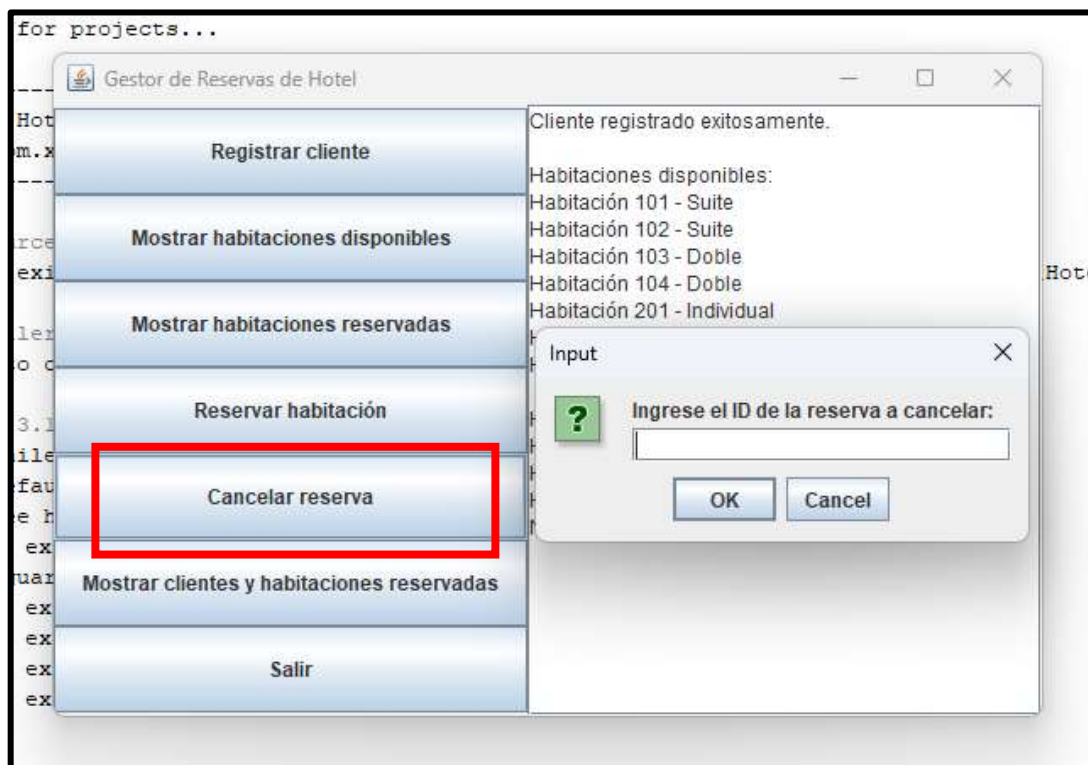
Mostrar habitaciones reservadas también nos da la opción de ver las habitaciones que ya cuentan con una reserva



Reservar habitación; nos deja interactuar con los datos que tenemos en habitaciones disponibles, y mediante esto podemos designar un cliente una habitación



Cancela reserva cumplirá la función de cancelar utilizando el id designado a la reserva



Mostrar clientes y habitaciones reservadas nos da un listado de la información solicitada



3.5 Manejo de Archivos

Tablas implementadas en la base de datos hotel_reservas



Contenido de la tabla clientes

✓ Mostrando filas 0 - 5 (total de 6, La consulta tardó 0,0010 segundos.)

```
SELECT * FROM `clientes`
```

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: Filtrar filas: Or

Opciones extra

			id_cliente	nombre	ci	nro_celular	
<input type="checkbox"/>	Editar	Copiar	Borrar	1	luis	123	73523382
<input type="checkbox"/>	Editar	Copiar	Borrar	2	NULL	NULL	NULL
<input type="checkbox"/>	Editar	Copiar	Borrar	3	Juan Torrez	1512333	60007899
<input type="checkbox"/>	Editar	Copiar	Borrar	4	Jose Torrez	1512453	45454870
<input type="checkbox"/>	Editar	Copiar	Borrar	5	Juan Torrez2	1111111	1111111
<input type="checkbox"/>	Editar	Copiar	Borrar	6	NULL	NULL	NULL

☐ Seleccionar todo Para los elementos que están marcados: Editar Copiar

Contenido de la tabla habitaciones

✓ Mostrando filas 0 - 9 (total de 10, La consulta tardó 0,0007 segundos.)





`SELECT * FROM `habitaciones``

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: Filtar filas: Ordenar seg

Opciones extra

		id_habitacion	numero	tipo	precio	reservada
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	101 Suite	120 0
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2	102 Suite	120 0
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	3	103 Doble	80 0
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	4	104 Doble	80 0
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	5	105 Doble	80 1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	6	201 Individual	50 0
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	7	202 Individual	50 0
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	8	203 Individual	50 1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	9	204 Suite	130 1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	10	205 Doble	85 0

 ☐ Seleccionar todo Para los elementos que están marcados:  Editar  Copiar  Borrar

4. Pruebas del Sistema

Registrar cliente: solicitamos los datos necesarios para realizar el registro correspondiente

Ilustración 2: Registro del cliente, nombre



Ilustración 1: Registro del clientecito

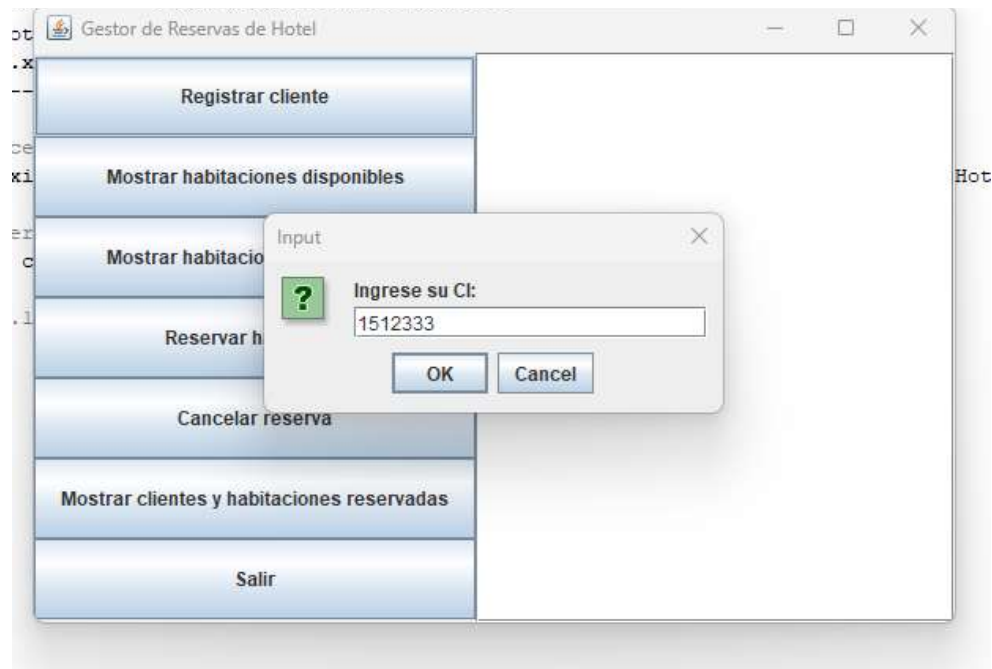


Ilustración 3: Registro del cliente, número de celular

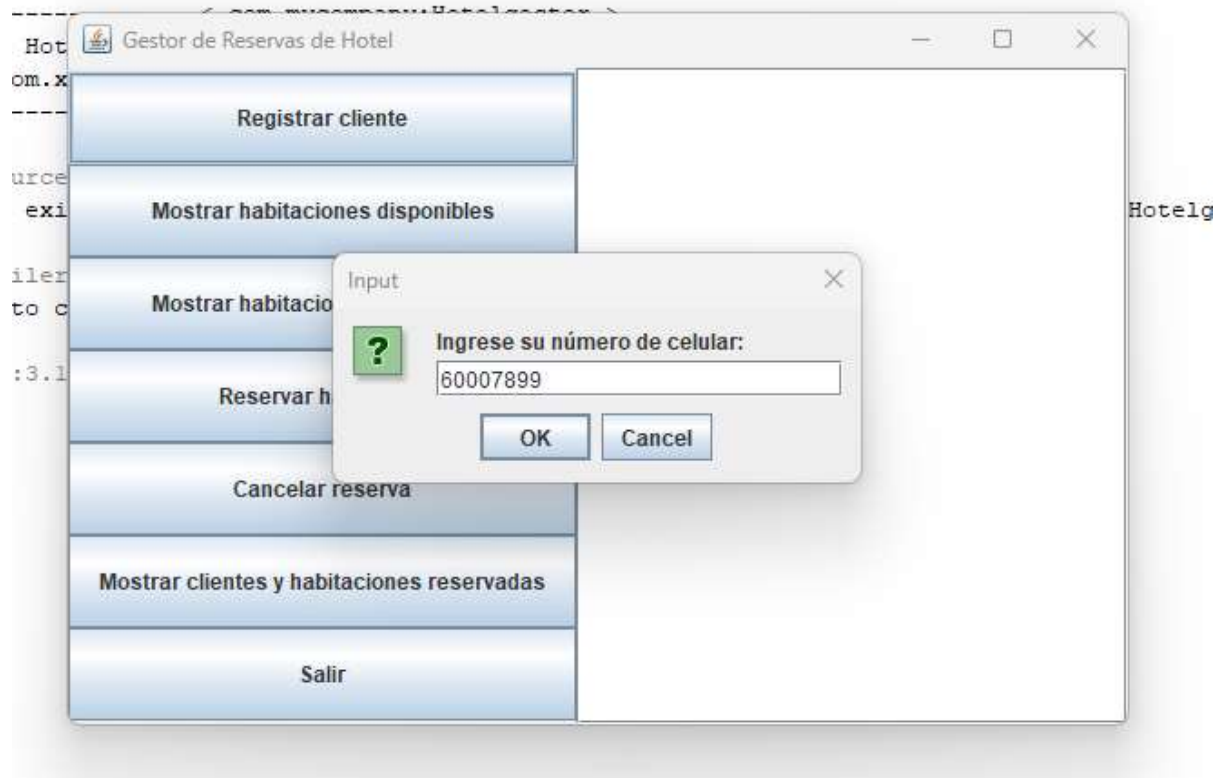
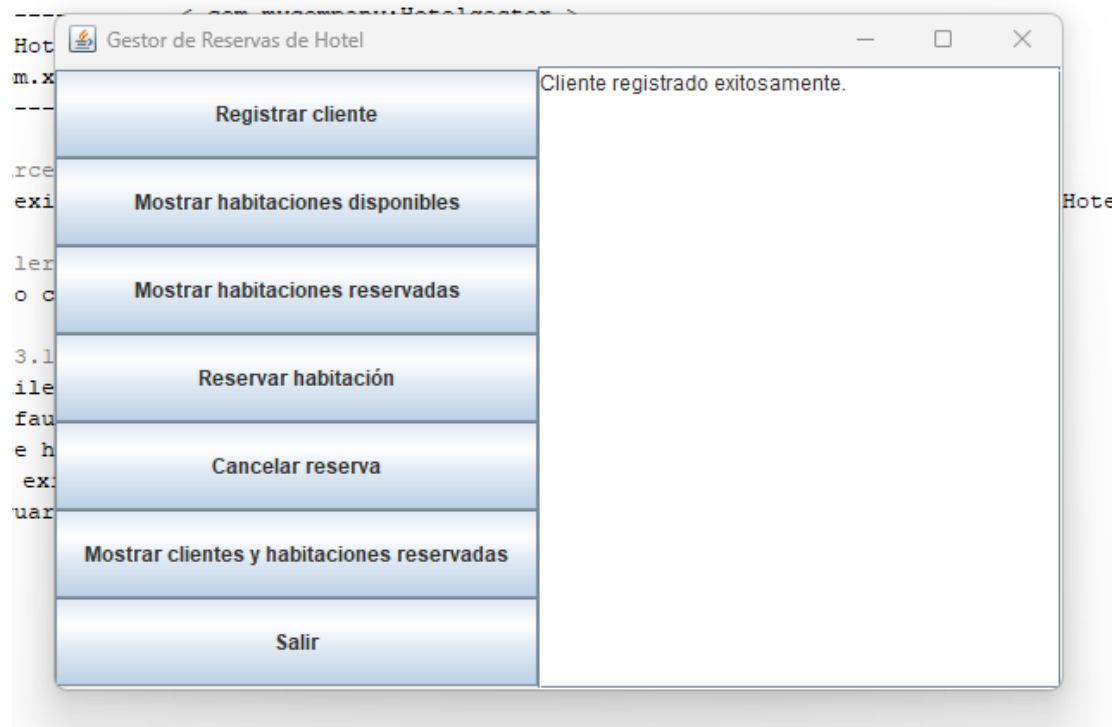
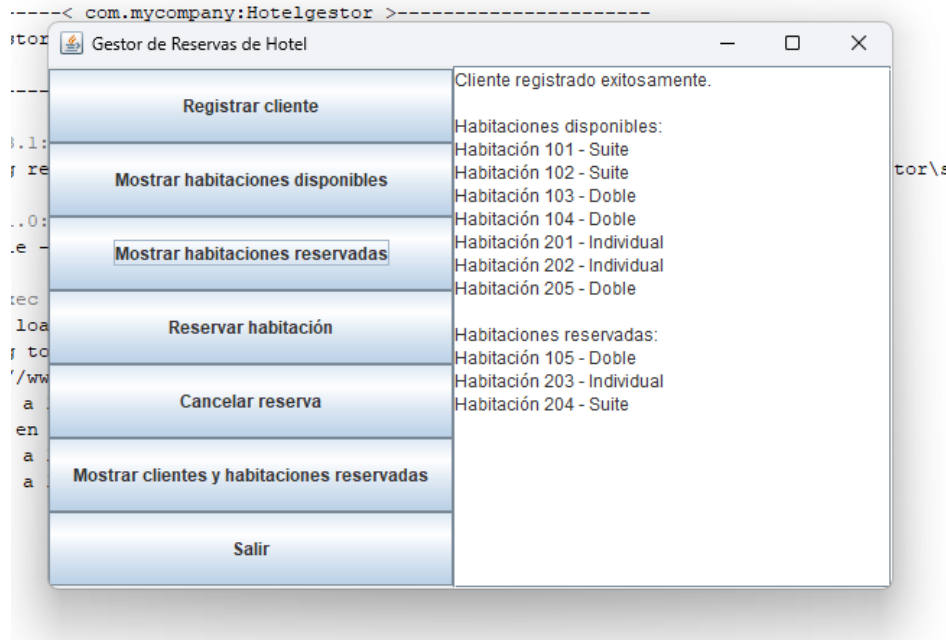


Ilustración 4 Verificación del registro exitoso



Verificamos habitaciones disponibles y reservadas para realizar la reserva de alguna de las disponibles

Ilustración 5: lista de habitaciones disponibles y reservadas



En base al listado mostrado podemos ingresar una nueva reserva, donde nos solicitaran los siguientes datos; la habitación, la fecha de inicio de la reserva y a la fecha de finalización y también solicitamos el id del cliente para realizar la reserva

Ilustración 6 número de habitación disponible para reservar



Ilustración 7 fecha de inicio de la reserva

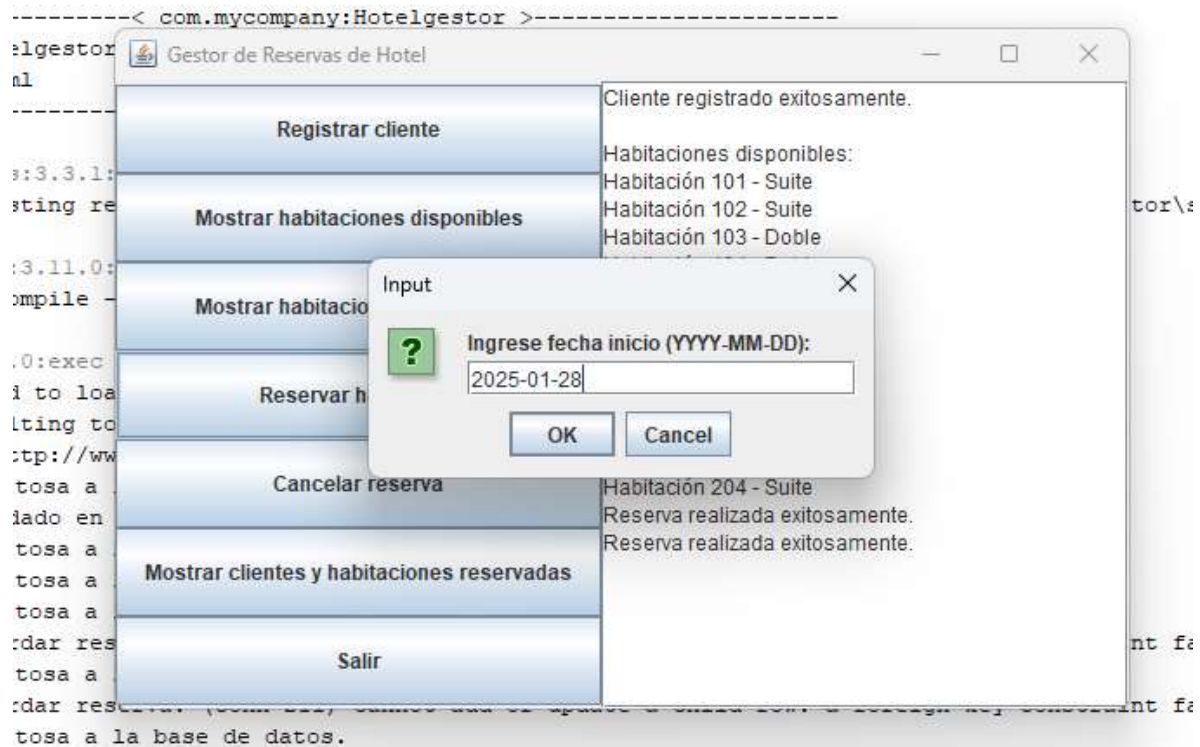


Ilustración 8 fecha de finalización de la reserva

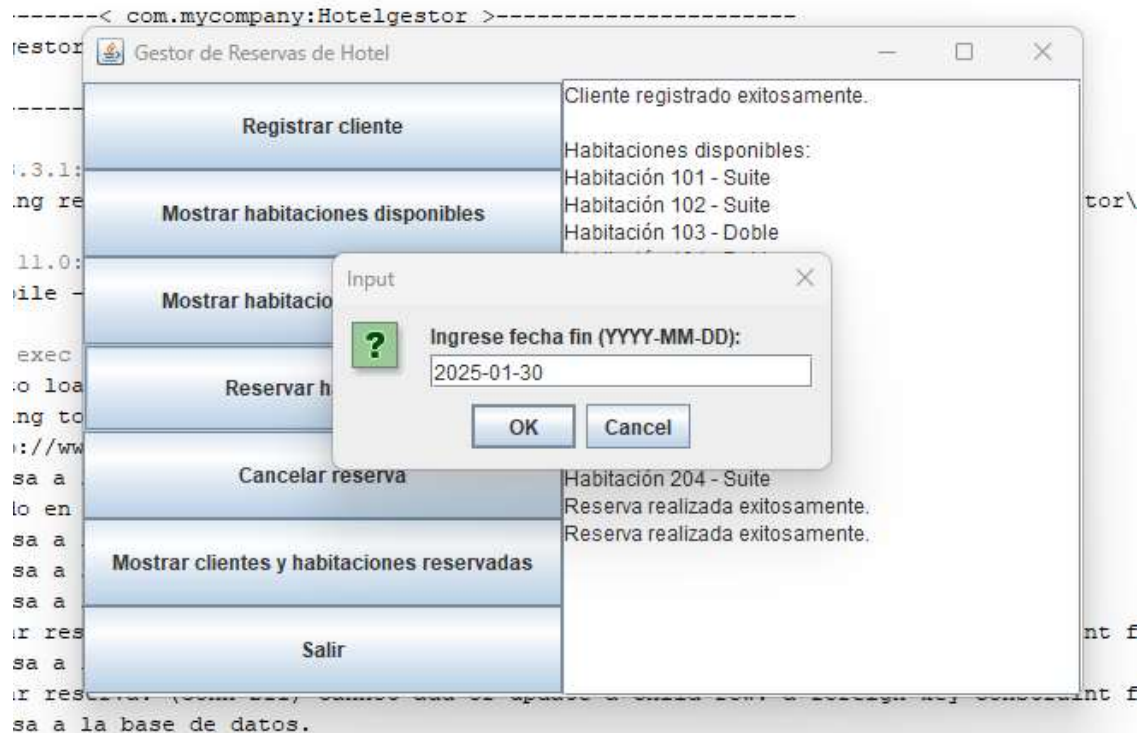
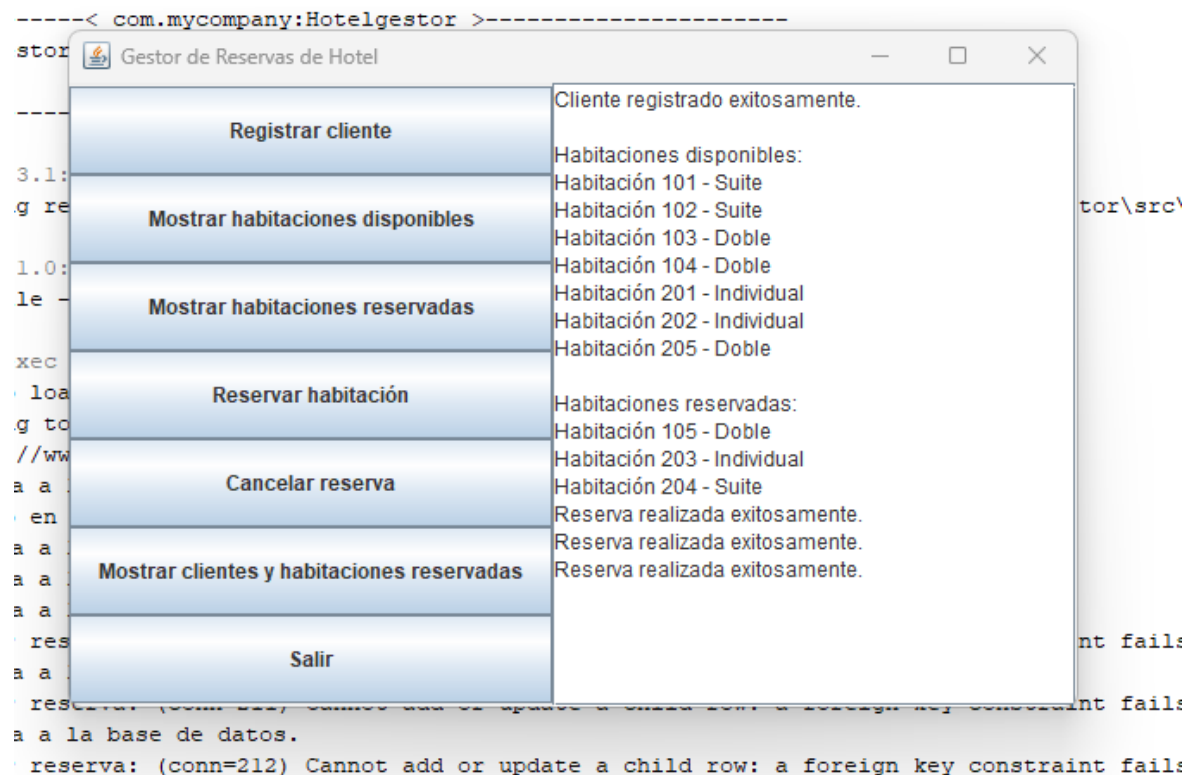
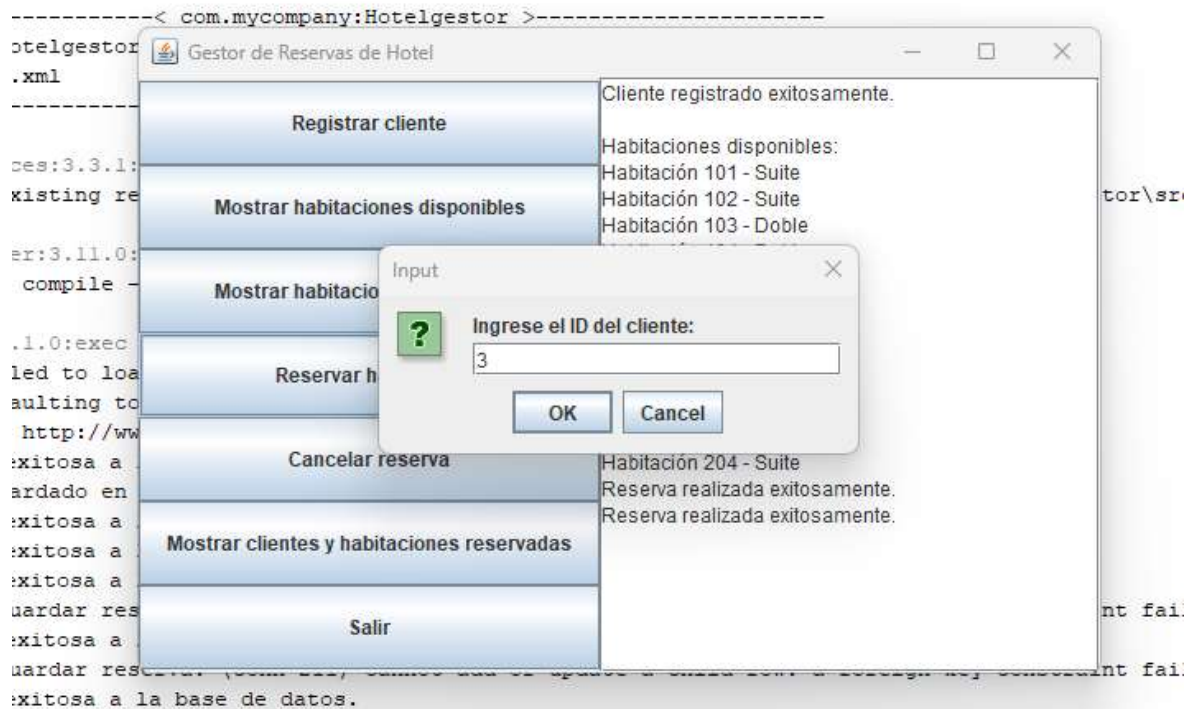
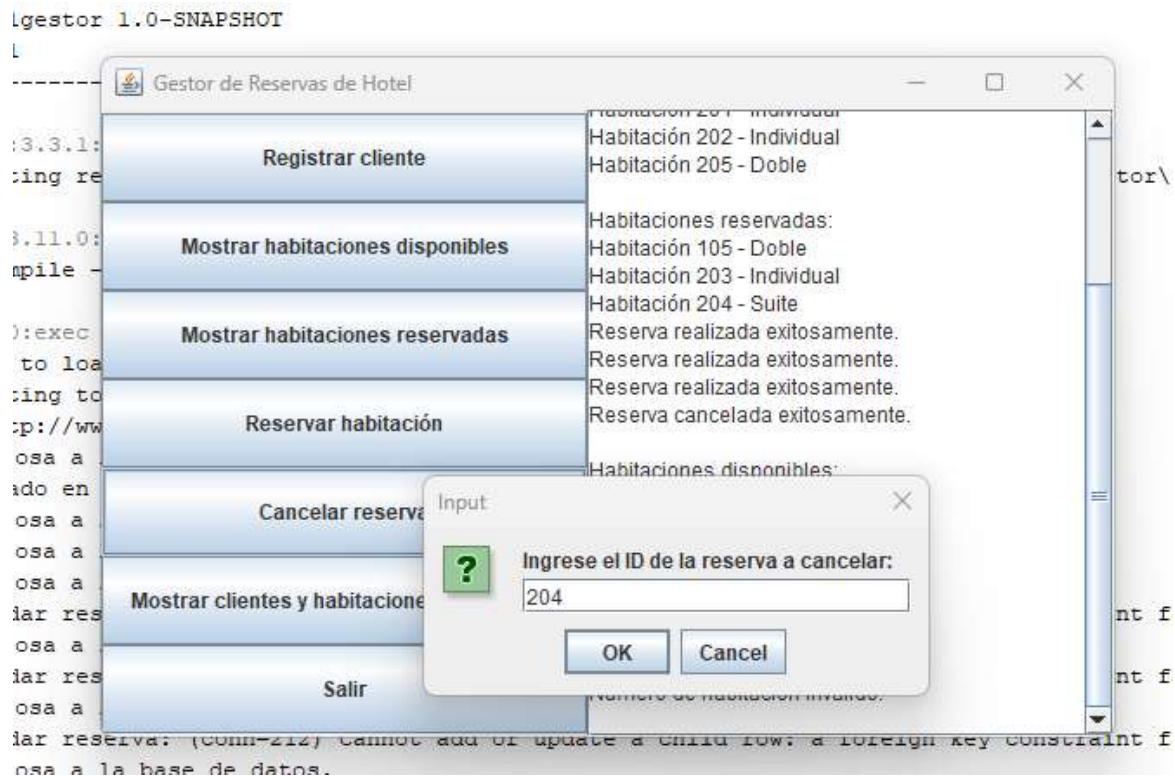


Ilustración 9 Id del cliente para la reserva



También podemos cancelar alguna reserva

Ilustración 10 Cancelación de reserva usando el id de la habitación



5. Conclusión

El proyecto demuestra la aplicación efectiva de los principios de POO en el desarrollo de un sistema funcional y escalable. La separación de responsabilidades y el uso de patrones de diseño permiten que el sistema sea mantenible y flexible ante futuras necesidades. Además, la implementación modular facilita su adaptación para soportar nuevas funcionalidades o integrar tecnologías adicionales.