

Reporte: Implementación del Patrón Facade en el Sistema de Autenticación y Registro

Introducción

En el desarrollo de nuestro sistema de citas médicas, nos enfrentamos al desafío de manejar la autenticación y el registro de usuarios de manera eficiente. Para simplificar la interacción con los subsistemas de base de datos y validaciones, decidimos implementar el **patrón Facade**. Este patrón nos permite proporcionar una interfaz simplificada para realizar operaciones complejas, como iniciar sesión y registrar nuevos usuarios.

¿Qué es el Patrón Facade?

El patrón **Facade** es un patrón de diseño estructural que proporciona una interfaz simplificada para un conjunto de interfaces en un subsistema. Este patrón es especialmente útil cuando tienes un sistema complejo con muchas clases y dependencias, y quieres ocultar esa complejidad detrás de una interfaz sencilla.

Participantes del Patrón Facade

En nuestra implementación, los participantes son:

1. FacadeAutenticacion (Fachada):

- Es la clase que proporciona una interfaz simplificada para iniciar sesión y registrar usuarios.
- Oculta la complejidad de las operaciones de base de datos y validaciones.

2. Subsistemas:

- **Gestión de Usuarios:** Realiza operaciones de base de datos para buscar y registrar usuarios.
- **Validaciones:** Verifica si un CURP, RFC o correo ya están registrados.

3. Cliente (Main):

- Es el código que utiliza la **Fachada** para realizar operaciones de autenticación y registro.

¿Cómo Funciona el Patrón Facade?

El patrón Facade funciona de la siguiente manera:

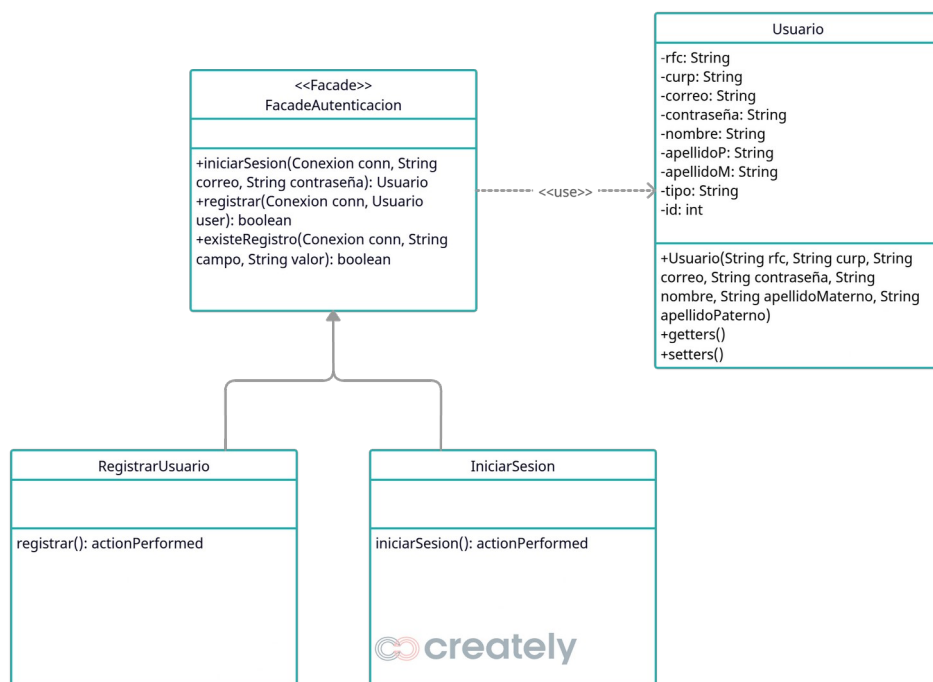
- 1.El **Ciente** (Main) interactúa con la **Fachada** (FacadeAutenticacion) para realizar operaciones de autenticación y registro.
- 2.La **Fachada** coordina las llamadas a los subsistemas (gestión de usuarios y validaciones) para realizar las operaciones necesarias.
- 3.El **Ciente** no necesita conocer los detalles de implementación de los subsistemas, lo que simplifica su código.

Implementación en el Proyecto

Clases Involucradas

- 1.**FacadeAutenticacion**: Proporciona métodos simplificados para iniciar sesión y registrar usuarios.
- 2.**Usuario**: Representa a un usuario en el sistema.
- 3.**Main**: Cliente que utiliza la **Fachada** para realizar operaciones de autenticación y registro.

Diagrama UML



Código de Implementación

1. FacadeAutenticacion

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class FacadeAutenticacion {
    private Connection conn;

    public FacadeAutenticacion(Connection conn) {
        this.conn = conn;
    }

    // Método para iniciar sesión
    public Usuario iniciarSesion(String correo, String contraseña) {
        String sqlUsuario = "SELECT * FROM usuarios WHERE correo = ? AND contraseña = ?";

        try (PreparedStatement stmtUsuario =
            conn.prepareStatement(sqlUsuario)) {
            stmtUsuario.setString(1, correo);
            stmtUsuario.setString(2, contraseña);

            try (ResultSet rs = stmtUsuario.executeQuery()) {
                if (rs.next()) {
                    int id = rs.getInt("id");
                    String nombre = rs.getString("nombre");
                    String aP = rs.getString("apellido_p");
                    String aM = rs.getString("apellido_m");
                    String rfc = rs.getString("rfc");
                    String curp = rs.getString("curp");
                    String tipo = rs.getString("tipo");

                    Usuario usuario = new Usuario(rfc, curp, correo,
                        contraseña, nombre, aM, aP);
                    usuario.setTipo(tipo);
                    usuario.setId(id);

                    System.out.println("Usuario encontrado: " +
                        usuario.getCurp());
                    return usuario;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        JOptionPane.showMessageDialog(null, "Correo o contraseña incorrectos", "Error", JOptionPane.ERROR_MESSAGE);
        return null;
    }

    // Método para registrar un nuevo usuario
}
```

```

public boolean registrar(Usuario usuario) {
    try {
        // Verificar si el CURP, RFC o correo ya están registrados
        if (existeRegistro("curp", usuario.getCurp())) {
            JOptionPane.showMessageDialog(null, "La CURP ya ha sido
registrada en otro usuario", "Advertencia", JOptionPane.WARNING_MESSAGE);
            return false;
        } else if (existeRegistro("rfc", usuario.getRfc())) {
            JOptionPane.showMessageDialog(null, "El RFC ya ha sido
registrado en otro usuario", "Advertencia", JOptionPane.WARNING_MESSAGE);
            return false;
        } else if (existeRegistro("correo", usuario.getCorreo())) {
            JOptionPane.showMessageDialog(null, "El correo ya ha sido
registrado en otro usuario", "Advertencia", JOptionPane.WARNING_MESSAGE);
            return false;
        } else {
            // Insertar el nuevo usuario
            String sql = "INSERT INTO Usuarios (rfc, curp, correo,
contraseña, nombre, apellido_p, apellido_m, tipo) VALUES
(?, ?, ?, ?, ?, ?, ?, ?)";
            try (PreparedStatement stmt = conn.prepareStatement(sql))
            {
                stmt.setString(1, usuario.getRfc());
                stmt.setString(2, usuario.getCurp());
                stmt.setString(3, usuario.getCorreo());
                stmt.setString(4, usuario.getContraseña());
                stmt.setString(5, usuario.getNombre());
                stmt.setString(6, usuario.getApellidoPaterno());
                stmt.setString(7, usuario.getApellidoMaterno());
                stmt.setString(8, "usuario"); // Tipo de usuario por
defecto

                int filasInsertadas = stmt.executeUpdate();
                if (filasInsertadas > 0) {
                    JOptionPane.showMessageDialog(null, "Registro
realizado correctamente", "Éxito", JOptionPane.INFORMATION_MESSAGE);
                    return true;
                }
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Problemas al realizar el
registro", "Error", JOptionPane.ERROR_MESSAGE);
        e.printStackTrace();
    }

    return false;
}

// Método auxiliar para verificar si un campo ya está registrado
private boolean existeRegistro(String campo, String valor) throws
SQLException {
    String sql = "SELECT * FROM usuarios WHERE " + campo + " = ?";
    try (PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, valor);
        try (ResultSet rs = stmt.executeQuery()) {
            return rs.next(); // Retorna true si existe un registro
con ese valor

```

```

    }
}
}

```

Clase Usuario

```

public class Usuario {
    private int id;
    private String rfc;
    private String curp;
    private String correo;
    private String contraseña;
    private String nombre;
    private String apellidoMaterno;
    private String apellidoPaterno;
    private String tipo;

    public Usuario(String rfc, String curp, String correo, String
contraseña, String nombre, String apellidoMaterno, String
apellidoPaterno) {
        this.rfc = rfc;
        this.curp = curp;
        this.correo = correo;
        this.contraseña = contraseña;
        this.nombre = nombre;
        this.apellidoMaterno = apellidoMaterno;
        this.apellidoPaterno = apellidoPaterno;
    }

    // Getters y Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getRfc() {
        return rfc;
    }

    public String getCurp() {
        return curp;
    }

    public String getCorreo() {
        return correo;
    }

    public String getContraseña() {
        return contraseña;
    }

    public String getNombre() {
        return nombre;
    }
}

```

```

    public String getApellidoMaterno() {
        return apellidoMaterno;
    }

    public String getApellidoPaterno() {
        return apellidoPaterno;
    }

    public String getTipo() {
        return tipo;
    }

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

```

Cliente IniciarSesion

```

private void iniciar_SesionMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    String correoText = correo.getText();
    String contraseñaText = new String(contraseña.getPassword());
    if(correoText.isEmpty() || contraseñaText.isEmpty()){
        JOptionPane.showMessageDialog(this, "Rellena todos los campos", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }else{
        Usuario usuario =
FacadeAutenticacion.iniciarSesion(conn,correoText,contraseñaText);
        if (usuario==null){
            JOptionPane.showMessageDialog(this, "El usuario y/o contraseña son incorrectos", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }else if(usuario.getTipo().equals("usuario")){
            JOptionPane.showMessageDialog(null,"Ha iniciado sesion");
            this.setVisible(false);
            Inicio_Usuario inicio = new Inicio_Usuario(usuario,conn);
            inicio.setVisible(true);
        } else if(usuario.getTipo().equals("medico_general")){
            JOptionPane.showMessageDialog(null,"Ha iniciado sesion");
            this.setVisible(false);
            Vista_Doctor inicio = new Vista_Doctor(usuario,conn);
            inicio.setVisible(true);
        }
    }
}

```

Cliente Registrar

```

private void RegistrarseMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    String rfcText = rfc.getText();
    String curpText = curp.getText();
    String Nombre = nombre.getText();
    String correoText = correo.getText();
    String contraseñaText = new String(contraseña.getPassword());

```

```

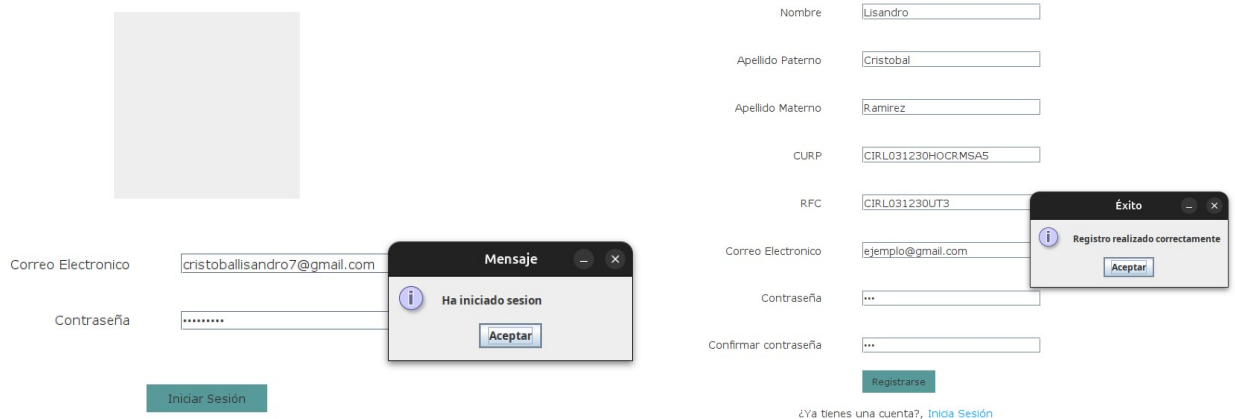
        String confirmarContraseñaText = new
String(confirmar_contraseña.getPassword());
        String apellidoMaternoText = apellido_m.getText();
        String apellidoPaternoText = apellido_p.getText();
        if (Nombre.length()==0 || correoText.length()==0 ||
contraseñaText.length()==0 || confirmarContraseñaText.length()==0 ||
apellidoMaternoText.length()==0 ||apellidoPaternoText.length()==0 ){
            JOptionPane.showMessageDialog(this, "Rellena todos los
campos", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        if(validarCurp(curpText) && validarRFC(rfcText)){

            if (!contraseñaText.equals(confirmarContraseñaText)) {
                JOptionPane.showMessageDialog(this, "Las contraseñas no
coinciden", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Crear el objeto Usuario si la contraseña coincide
            Usuario usuario = new Usuario(rfcText, curpText, correoText,
contraseñaText, Nombre, apellidoMaternoText, apellidoPaternoText);
            if(FacadeAutenticacion.registrar(conn, usuario)){
                Iniciar_Sesion iniciar = new Iniciar_Sesion(conn);
                iniciar.setVisible(true);
                this.setVisible(false);
            }
        }
    }
}

```

Salida del Código



The image displays two screenshots of a web application interface. The left screenshot shows a login form with fields for 'Correo Electronico' (cristoballisandro7@gmail.com), 'Contraseña' (masked with dots), and a green 'Iniciar Sesión' button. A modal message box titled 'Mensaje' is overlaid, stating 'Ha iniciado sesion' with an 'Aceptar' button. The right screenshot shows a registration form with fields for 'Nombre' (Luisandro), 'Apellido Paterno' (Cristobal), 'Apellido Materno' (Ramirez), 'CURP' (CIRL031230HOCRMSAS), 'RFC' (CIRL031230UT3), 'Correo Electronico' (ejemplo@gmail.com), 'Contraseña' (masked), and 'Confirmar contraseña' (masked). A green 'Registrarse' button is at the bottom. A modal message box titled 'Éxito' is overlaid, stating 'Registro realizado correctamente' with an 'Aceptar' button. Below the registration form, there is a link: '¿Ya tienes una cuenta?, [Inicia Sesión](#)'.

Conclusión

El patrón Facade ha sido una excelente adición a nuestro sistema de citas médicas. Nos ha permitido simplificar la interacción con los subsistemas de autenticación y registro, ocultando la complejidad detrás de una interfaz sencilla. Esto hace que el código sea más fácil de usar, mantener y extender.