

Entrega Parcial - Proyecto Base de Datos – Fase Final

Grupo del Proyecto No. 5

Fecha: 11 de agosto de 2025

1. Nombre del Proyecto

ClinicData.

2. Nombre de la Aplicación

Sistema Digital para Clínica.

3. Objetivo General

Desarrollar un sistema digital de gestión clínica que mejore la administración de pacientes, doctores, citas médicas y reportes en una clínica. El sistema busca reemplazar el uso excesivo de archivos físicos con un entorno digital organizado, seguro y accesible, optimizando el flujo de trabajo dentro del entorno clínico y mejorando la eficiencia en la atención al paciente.

4. Funcionalidades del Sistema

1. Registro y gestión de pacientes - Completada

```
public static void guardarRegistro(String cedula, String nombre, String apellido, int edad, String correo, String telefono,
|                                String direccion1, String usuario, String contrasena, String rol){
|    String hashedPassword = BCrypt.hashpw(contrasena, BCrypt.gensalt());
|    String sql = "INSERT INTO personas (cedula, nombre, apellido, edad, correo, telefono, direccion1, usuario, contraseña,
|    " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
|    try(Connection connection = DatabaseConnection.getConnection();
|        PreparedStatement statement = connection.prepareStatement(sql)){
|        statement.setString(parameterIndex:1, cedula);
|        statement.setString(parameterIndex:2, nombre);
|        statement.setString(parameterIndex:3, apellido);
|        statement.setInt(parameterIndex:4, edad);
|        statement.setString(parameterIndex:5, correo);
|        statement.setString(parameterIndex:6, telefono);
|        statement.setString(parameterIndex:7, direccion1);
|        statement.setString(parameterIndex:8, usuario);
|        statement.setString(parameterIndex:9, hashedPassword);
|        statement.setString(parameterIndex:10, rol);
|
|        statement.executeUpdate();
|        System.out.println("Registro Guardado Exitosamente.");
|
|    }catch (SQLException e){
|        System.out.println("error al guardar el registro: " + e.getMessage());
|        e.printStackTrace();
|    }
|}
```

Result Grid										
	cedula	nombre	apellido	edad	correo	telefono	direccion1	usuario	contraseña	rol
▶	0912345678	Juan	Pérez	45	juan.perez@hospital.com	0987654321	Av. Principal 123	jperez	\$2y\$10\$92ZXUNpkjO0rOQ5byMl.Ye4okCeBa3Ro9lC/.og...	admin
	091239456	chris	si	20	christian@hospital.com	0943905968	alborada	chris123	\$2a\$10\$DSzrW6nMaxFlc08N2tad.XSf6kh7f6s/4bx8...	cliente
	0923456789	María	González	38	maria.gonzalez@hospital.com	0976543210	Calle Secundaria 456	mgonz...	\$2y\$10\$Tdh8H1.PfQx37tgCzwikb.KJNyWgahB9cbcoQ...	admin
	0932038797	sebas	villa	19	sebas@gmail.com	0967023910	alborada 3ra	sevilla	\$2a\$10\$fm/RfQzPBOK/zdDGx302uIPXzxC/BfTmrAL9...	admin
	0934567890	Carlos	Rodríguez	42	carlos.rodriguez@hospital.com	0965432109	Av. Médica 789	crodrig...	\$2y\$10\$g9FqIX7d.o4QJKGH2k.9c8FoFf.f.TG9pq...	staff
	0945678901	Ana	Martínez	35	ana.martinez@hospital.com	0954321098	Calle Doctor 3	amartinez	\$2y\$10\$98GrYm8e.p5RklH13l.0dg9PrGgMg.uHc0qr...	staff
	0955585187	paula	burgos	20	pau.bn@gmail.com	093909936	31 y maldonado	pauvn	\$2a\$10\$94gluR8EMvUtxgW/z2key7JQn930125h...	admin
	0956655781	ariana	bravo	23	arianadabra@gmail.com	0991505929	rumichaca1024	arienco	\$2a\$10\$9fRPQ14.1gdMo2C8pQlOHj6M6tDnkqgEY1...	admin
	0956789012	Luis	Fernández	29	luis.fernandez@hospital.com	0943210987	Av. Enfer 6	lfernand...	\$2y\$10\$93d4hZn9f.qQ65mMJ4m..1e0lQshHNh.vId1r...	enfermero
	0967890123	Carmen	López	31	carmen.lopez@hospital.com	0932109876	Calle Cuidados 9	dlopez	\$2y\$10\$H0elAt0ig.r7ThnKk5n.2fIRttIOi.wJe2sthp...	enfermero
	0978901234	Pedro	Silva	55	pedro.silva@gmail.com	0921098765	Urb Norte 1	psilva	\$2y\$10\$1fJubjh.s8UoOL6o.3g2SuUJP.JKF3tOUq5...	cliente
	0987654321	prueba	prueba	22	prueba@prueba.com	0123456789	prueba	prueba	\$2a\$10\$9CmPj5jdBPWuu4PjN2uh...	admin
	0989012345	Rosa	Herrera	48	rosa.herrera@outlook.com	0910987654	Sector Sur 2	rherrea	\$2y\$10\$32gKvCa2i.t79vpMm7p.4h3kTvKQk.yLg4uv...	cliente
	0990123456	Samuel	Villa	33	samuel.villa@hospital.com	0909876543	Calle Nueva 789	savilla	\$2y\$10\$9K3lwDr3J.uU0WqQNn8q.Si4luLIRJ.zMh5vw...	admin
	2450125832	christian	olmedo	19	chriolme@espol.edu.ec	0999487402	santa elena	chris	\$2a\$10\$NCJmhYuecv6LnnS3p67euFBuclTHA0c1fGB1f...	admin
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

 Clinic data | agregar usuario

Registrar un usuario nuevo

Cédula:

Nombre:

Apellido:

Edad:

Correo:

Teléfono:

Dirección:

Usuario:

Contraseña:

Rol:

2. Administración del personal clínico

Administrador - eliminar un usuario

Usuario de la persona a eliminar:

RECUEDE: ESTA ACCIÓN ES IRREVERSIBLE, PROCEDA CON DUDADO.

```
package com.espol.proyectodb3;

> import ...  

public class EliminarUsuarioController { & Samuel Villagomez  

    @FXML 1 usage  

    private Button btn_eliminarUsuario, btn_cancelar;  

    @FXML  

    private TextField txt_usuarioAeliminar;  

    @FXML  

    private Label lbl_msg;  

    //cancelar y volver al panel de admin  

    @FXML 1 usage & Samuel Villagomez  

    public void cancelarEliminacion(ActionEvent event){  

        //volver a la ventana de admin  

        try {  

            lbl_msg.setText("Cancelando...");  

            PauseTransition delay = new PauseTransition(Duration.seconds( 2 ));  

            delay.setOnFinished( ActionEvent e -> {  

                try {  

                    Parent root = FXMLLoader.load(getClass().getResource( name: "Admin-view.fxml" ));  

                    Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  

                    stage.setScene(new Scene(root));  

                    stage.show();  

                    stage.centerOnScreen();  

                } catch (IOException ex) {  

                    ex.printStackTrace();  

                }  

            });
            delay.play();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

//eliminar el usuario  

@FXML 1 usage & Samuel Villagomez  

public void eliminarUsuario(){  

    if (txt_usuarioAeliminar.getText() == "" || txt_usuarioAeliminar.getText().isEmpty()){  

        lbl_msg.setText("Error, rellene todos los campos.");  

    } else {  

        String usuario = txt_usuarioAeliminar.getText();  

        UserValidations validator = new UserValidations();  

        validator.eliminarUsuario(usuario, lbl_msg);  

        Stage stage = (Stage) btn_eliminarUsuario.getScene().getWindow();  

        PauseTransition delay = new PauseTransition(Duration.seconds( 1 ));  

        delay.setOnFinished( ActionEvent event -> stage.close());  

        delay.play();
    }
}
```

3. Programación de citas médicas

Agendar Nueva Cita

Seleccionar Doctor:

Seleccione un doctor...

Fecha de la Cita:

Seleccione una fecha... 

Hora de la Cita:

Seleccione una hora...

Descripción del motivo:

Describa el motivo de su consulta...

Agendar Cita

Cancelar

```
9
10 public class NuevaCitaController {
11
12     @FXML
13     private DatePicker datePicker;
14     @FXML
15     private ComboBox<String> cbSpecialty;
16     @FXML
17     private ComboBox<String> cbDoctor;
18     @FXML
19     private TextArea taDescription;
20
21     public void handleCancel(ActionEvent event) { 1 usage
22         try {
23             //cerrar
24             javafx.scene.Node source = (javafx.scene.Node) event.getSource();
25             javafx.stage.Stage stage = (javafx.stage.Stage) source.getScene().getWindow();
26             stage.close();
27         } catch (Exception e) {
28             e.printStackTrace();
29             System.out.println("Error al cerrar la ventana de nueva cita.");
30         }
31     }
32 }
33 }
```

4. Consultar, modificar o cancelar citas

Clinic data | panel de personal

Panel Médico

Cerrar sesión

Citas

Pacientes

Recetas

Citas cargadas: 3

Fecha	Paciente	Motivo	Estado	Acciones
2025-08-25	pruebClient43 pruebClien...	assasd	cancelada	<button>Ver Detalles</button>
2025-08-25	pruebClient43 pruebClien...	asdad	pendiente	<button>Ver Detalles</button>
2025-08-25	pruebClient43 pruebClien...	wkdflksjdf	completada	<button>Ver Detalles</button>

Clinic data | panel de personal

Panel Médico

Cerrar sesión

Gestión de Cita

Cita de pruebClient43 pruebClient43

Fecha: 2025-08-25

Hora: 14:00:00

Departamento: General

Descripción: asdad

Cambiar Estado: pendiente

Selección el nuevo estado y presiona 'Actualizar'

Actualizar Estado

Cerrar

Gestión de Cita

Cita de pruebClient43 pruebClient43

Fecha: 2025-08-25

Hora: 14:00:00

Departamento: General

Descripción: asdad

Cambiar Estado:

Selecciona el nuevo estado para la cita.

Actualizar

Actualizar Estado

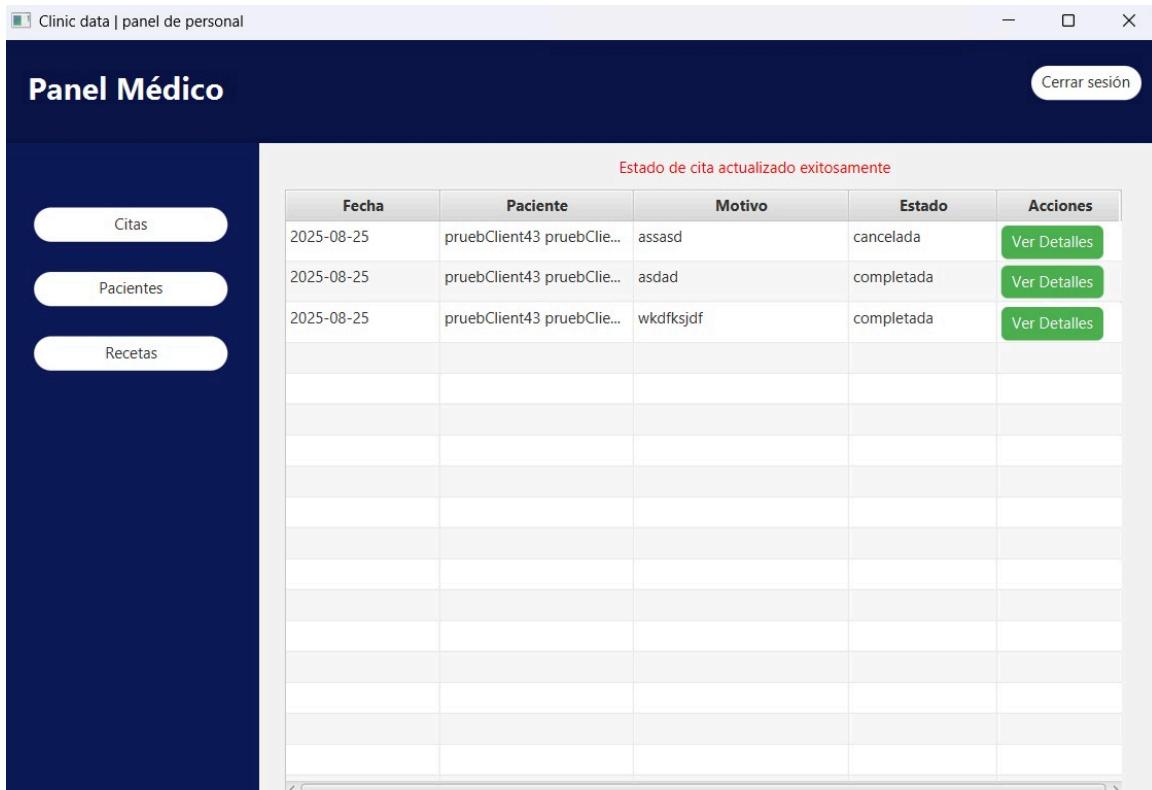
Cerrar

Estado Actualizado

Actualización Exitosa

El estado de la cita de pruebClient43 pruebClient43 se actualizó a: COMPLETADA

Aceptar



```
public class ClientController implements Initializable { & Chrisxho2006 +1
    private void obtenerCedulaCliente(String usuario) { 1 usage & Chrisxho2006
    }

    @FXML 4 usages & Chrisxho2006
    private void cargarMisCitas() {
        if (cedulaCliente == null) {
            mostrarError(titulo: "Error", mensaje: "No se pudo obtener la información del cliente");
            return;
        }

        try {
            List<Cita.CitaClienteRow> citas = Cita.obtenerCitasPorCliente(cedulaCliente);
            citasList.clear();
            citasList.addAll(citas);

        } catch (SQLException e) {
            mostrarError(titulo: "Error al cargar citas", e.getMessage());
            e.printStackTrace();
        }
    }

    private void eliminarCitaSeleccionada(Cita.CitaClienteRow cita) { 1 usage & Chrisxho2006
        // Usar Platform.runLater para el diálogo de confirmación
        javafx.application.Platform.runLater(() -> {
            Alert confirmacion = new Alert(Alert.AlertType.CONFIRMATION):

```

```

private void eliminarCitaSeleccionada(Cita.CitaClienteRow cita) { 1 usage  & Chrisxho2006
    // Usar Platform.runLater para el diálogo de confirmación
    javafx.application.Platform.runLater(() -> {
        Alert confirmacion = new Alert(Alert.AlertType.CONFIRMATION);
        confirmacion.setTitle("Confirmar eliminación");
        confirmacion.setHeaderText("¿Está seguro que desea eliminar esta cita?");
        confirmacion.setContentText("Fecha: " + cita.getFechaFormatada() +
            "\nDoctor: " + cita.getDoctorCompleto() +
            "\nEspecialidad: " + cita.doctorEspecialidad);

        Optional<ButtonType> resultado = confirmacion.showAndWait();
        if (resultado.isPresent() && resultado.get() == ButtonType.OK) {
            try {
                boolean eliminada = Cita.eliminarCita(cita.citaId, cedulaCliente);
                if (eliminada) {
                    citasList.remove(cita);
                    mostrarInfo(titulo: "Éxito", mensaje: "Cita eliminada correctamente");
                } else {
                    mostrarError(titulo: "Error", mensaje: "No se pudo eliminar la cita");
                }
            } catch (SQLException e) {
                mostrarError(titulo: "Error al eliminar cita", e.getMessage());
                e.printStackTrace();
            }
        }
    });
}

```

```

public class ClientController implements Initializable { & Chrisxho2006+1
    @FXML 2 usages  & Chrisxho2006
    public void AgendarCita(javafx.event.ActionEvent event) {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource(name: "NuevaCita-view.fxml"));
            Parent root = loader.load();

            // Pasar información del cliente al controlador de nueva cita
            NuevaCitaController nuevaCitaController = loader.getController();
            if (nuevaCitaController != null) {
                nuevaCitaController.setCedulaCliente(cedulaCliente);
                nuevaCitaController.setUsuarioCliente(usuarioActual);
            }

            Stage stage = new Stage();
            stage.setTitle("Nueva Cita");
            stage.setScene(new Scene(root));
            stage.show();
            stage.centerOnScreen();

            // Recargar citas cuando se cierre la ventana
            stage.setOnHidden(WindowEvent e -> cargarMisCitas());

        } catch (IOException e) {
            mostrarError(titulo: "Error", mensaje: "No se pudo abrir la ventana de nueva cita");
        }
    }
}

```

```
public void cargarCitasDelDoctor() { 4 usages  ↗ Chrisxho2006 +2*
    try {
        if (cedulaDoctor == null) {
            if (lbl_msg != null) {
                lbl_msg.setText("Error: Cédula del doctor no establecida");
            }
            return;
        }
        var lista = Cita.obtenerCitasPorDoctor(cedulaDoctor);

        if (tableAppointments != null) {
            tableAppointments.setItems(FXCollections.observableArrayList(lista));

            if (lbl_msg != null) {
                lbl_msg.setText("Citas cargadas: " + lista.size());
            }
        }
    } catch (Exception e) {
        System.err.println("Error cargando citas: " + e.getMessage());
        e.printStackTrace();
        if (lbl_msg != null) {
            lbl_msg.setText("Error cargando citas: " + e.getMessage());
        }
    }
}
```

```
public class DoctorController implements Initializable { ↗ Chrisxho2006 +2*
    private void verDetallesCita(Cita.CitaRow cita) { 1 usage  ↗ Chrisxho2006
        Dialog<String> dialog = new Dialog<>();
        dialog.setTitle("Gestión de Cita");
        dialog.setHeaderText("Cita de " + cita.clienteNombre + " " + cita.clienteApellido);

        // Crear el contenido del diálogo
        GridPane grid = new GridPane();
        grid.setHgap(15);
        grid.setVgap(15);
        grid.setPadding(new Insets( 20,  20,  10,  10));

        // Información de la cita (solo lectura)
        grid.add(new Label( s: "📅 Fecha:"),  i: 0,  ii: 0);
        Label lblFecha = new Label(cita.fecha.toString());
        lblFecha.setStyle("-fx-font-weight: bold;");
        grid.add(lblFecha,  i: 1,  ii: 0);

        grid.add(new Label( s: "⌚ Hora:"),  i: 0,  ii: 1);
        Label lblHora = new Label(cita.hora.toString());
        lblHora.setStyle("-fx-font-weight: bold;");
        grid.add(lblHora,  i: 1,  ii: 1);

        grid.add(new Label( s: "📍 Departamento:"),  i: 0,  ii: 2);
        grid.add(new Label(cita.departamento),  i: 1,  ii: 2);
    }
}
```

```

public class DoctorController implements Initializable { & Chrisxho2006+2*
    private void verDetallesCita(Cita.CitaRow cita) { 1 usage & Chrisxho2006

        grid.add(new Label("Descripción:"), i: 0, ii: 3);
        TextArea txtDescripcion = new TextArea(cita.descripcion != null ? cita.descripcion : "Sin descripción");
        txtDescripcion.setEditable(false);
        txtDescripcion.setPrefRowCount(2);
        txtDescripcion.setPrefColumnCount(30);
        grid.add(txtDescripcion, i: 1, ii: 3);

        // ComboBox para cambiar estado
        grid.add(new Label("Cambiar Estado:"), i: 0, ii: 4);
        ComboBox<String> estadoCombo = new ComboBox<>();
        estadoCombo.getItems().addAll("pendiente", "en_curso", "completada", "cancelada");
        estadoCombo.setValue(cita.estado);
        estadoCombo.setStyle("-fx-font-size: 14px;");

        // Agregar colores según el estado
        estadoCombo.setOnAction(ActionEvent e -> {
            String selectedState = estadoCombo.getValue();
            switch (selectedState) {
                case "pendiente":
                    estadoCombo.setStyle("-fx-base: #FFF3CD; -fx-font-size: 14px;"); // Amarillo
                    break;
                case "en_curso":
                    estadoCombo.setStyle("-fx-base: #CCE5FF; -fx-font-size: 14px;"); // Azul
                    break;
                case "completada":
                    estadoCombo.setStyle("-fx-base: #A9F5D0; -fx-font-size: 14px;"); // Verde
                    break;
                case "cancelada":
                    estadoCombo.setStyle("-fx-base: #F0E68C; -fx-font-size: 14px;"); // Naranja
                    break;
            }
        });
    }
}

```

```

public class DoctorController implements Initializable { & Chrisxho2006+2*
    private void cambiarEstadoCita(int citaId, String nuevoEstado, String nombrePaciente) { 1 usage & Chrisxho2006

        boolean exitoso = Cita.actualizarEstadoCita(citaId, nuevoEstado);

        if (exitoso) {
            // Mostrar mensaje de éxito
            Alert confirmacion = new Alert(Alert.AlertType.INFORMATION);
            confirmacion.setTitle("Estado Actualizado");
            confirmacion.setHeaderText("Actualización Exitosa");
            confirmacion.setContentText("El estado de la cita de " + nombrePaciente +
                " se actualizó a: " + nuevoEstado.replace(target: "_", replacement: " ").toUpperCase());
            confirmacion.showAndWait();

            // Refrescar la tabla
            cargarCitasDelDoctor();

            if (lbl_msg != null) {
                lbl_msg.setText("Estado de cita actualizado exitosamente");
            }
        } else {
            // Mostrar mensaje de error
            Alert error = new Alert(Alert.AlertType.ERROR);
            error.setTitle("Error");
            error.setHeaderText("No se pudo actualizar");
            error.setContentText("Hubo un problema al actualizar el estado de la cita. Verifica que el ID de la cita es válido.");
            error.showAndWait();
        }
    }
}

```

```

public class DoctorController implements Initializable { @Chrisxho2006+2*
    private void cambiarEstadoCita(int citaId, String nuevoEstado, String nombrePaciente) {
    }

    public void cargarRecetasDelDoctor() { no usages @Chrisxho2006 +2
        try {
            if (cedulaDoctor != null) {
                var lista = Tratamiento.obtenerTratamientosPorDoctor(cedulaDoctor);
                // tblRecetas.setItems(FXCollections.observableArrayList(lista));
            }
        } catch (Exception e) {
            if (lbl_msg != null) {
                lbl_msg.setText("Error recetas: " + e.getMessage());
            }
        }
    }

    public void initializeLabel(String text) { no usages @Chrisxho2006
        if (lbl_msg != null) {
            lbl_msg.setText(text);
        }
    }

    // Método para refrescar las citas (puedes llamarlo desde otras ventanas)
    public void refrescarCitas() { cargarCitasDelDoctor(); }
}

```

5. Control de acceso diferenciado por rol

```

protected void handleLogin(ActionEvent event){
    String usuario = txtUsername.getText();
    String contrasenia = txtPassword.getText();
    String hashedPassword = Utileria.encriptarContrasena(contrasenia);

    if (UserValidations.validateUser(usuario, contrasenia, role:"admin")){ //validar y saltar a la vista de admin
        try {
            root = FXMLLoader.load(getClass().getResource(name:"Admin-view.fxml"));
            stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            scene = new Scene(root);
            stage.setScene(scene);
            stage.setTitle("Clinic data | panel de administrador");
            stage.show();
        } catch (IOException e) {
            LabelMsg.setText("Usuario o contraseña incorrectos");
            e.printStackTrace();
        }
    } else if (UserValidations.validateUser(usuario, contrasenia, role:"staff")){ //validar y saltar a la vista de staff
        try {
            root = FXMLLoader.load(getClass().getResource(name:"Staff-view.fxml"));
            stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            scene = new Scene(root);
            stage.setScene(scene);
            stage.setTitle("Clinic data | panel de personal");
            //centrar la ventana
            stage.centerOnScreen();
            stage.show();
        } catch (IOException e) {
            LabelMsg.setText("Usuario o contraseña incorrectos");
            e.printStackTrace();
        }
    }
}

```

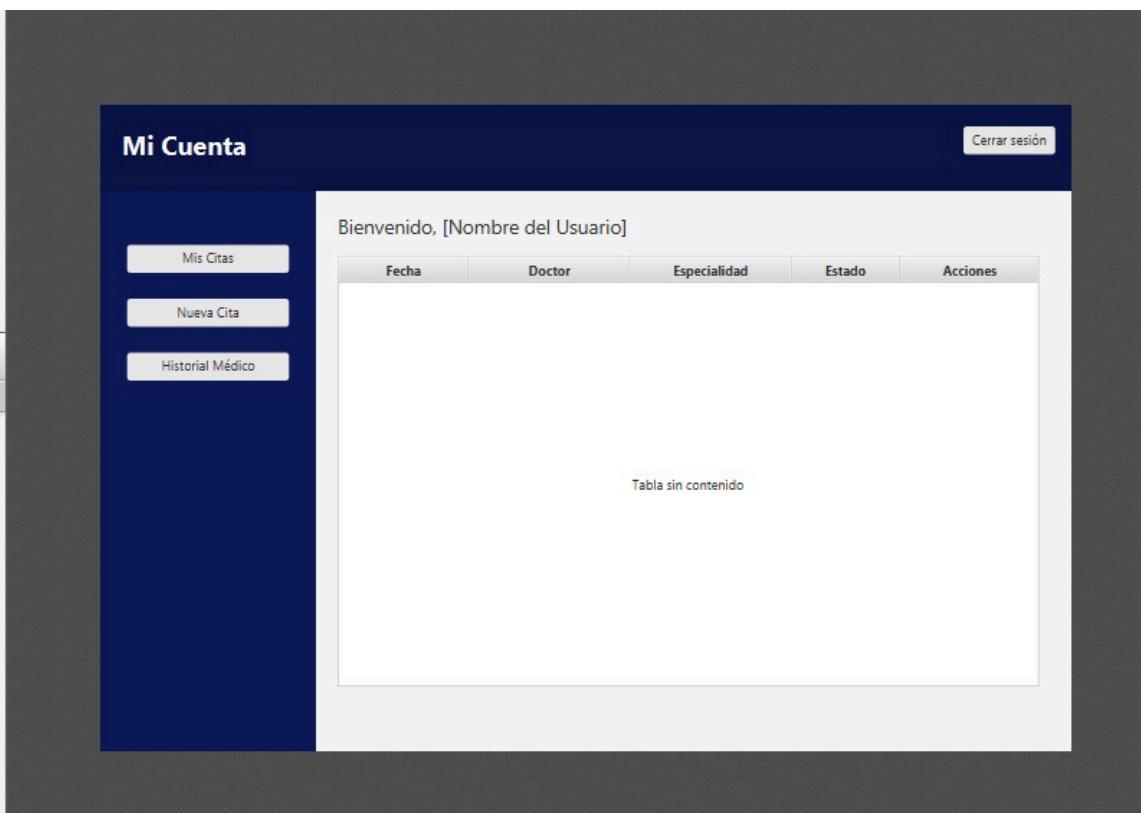
```

} else if (UserValidations.validateUser(usuario, contrasenia, role:"client")){ //validar y saltar a la vista de cliente
    try {
        root = FXMLLoader.load(getClass().getResource(name:"Client-view.fxml"));
        stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("Clinic data | panel de cliente");
        //centrar la ventana
        stage.centerOnScreen();
        stage.show();
    } catch (IOException e) {
        LabelMsg.setText("Usuario o contraseña incorrectos");
        e.printStackTrace();
    }
} else {
    LabelMsg.setText("Usuario o contraseña incorrectos");
}

```

Nota: el ejecutable será hecho en el video.

Vista cliente



Vista admin

The screenshot shows the 'Panel de Administración' (Admin Panel) interface. At the top right is a 'Cerrar sesión' (Logout) button. On the left, there's a sidebar with three 'Agregar un usuario' (Add User) buttons. The main area features a table with columns: ID, Nombre (Name), Rol (Role), Email, and Acciones (Actions). A message 'Tabla sin contenido' (Table empty) is displayed below the table.

Vista del personal

The screenshot shows the 'Panel Médico' (Medical Staff Panel) interface. At the top right is a 'Cerrar sesión' (Logout) button. On the left, there's a sidebar with three buttons: 'Citas' (Appointments), 'Pacientes' (Patients), and 'Recetas' (Prescriptions). The main area features a table with columns: Fecha (Date), Paciente (Patient), Motivo (Reason), Estado (Status), and Acciones (Actions). A message 'Tabla sin contenido' (Table empty) is displayed below the table.

5. Modelo Físico del Sistema

```
-- Tabla de usuarios generales
CREATE TABLE personas(
    cedula VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(15) NOT NULL,
    apellido VARCHAR(15) NOT NULL,
    edad INT NOT NULL,
    correo VARCHAR(30) NOT NULL UNIQUE,
    telefono VARCHAR(10) NOT NULL,
    direccion1 VARCHAR(20) NOT NULL,
    usuario VARCHAR(10) NOT NULL UNIQUE,
    contraseña VARCHAR(100) NOT NULL, -- recomendable
    encriptada
        rol ENUM('admin', 'staff', 'enfermero', 'cliente') NOT
    NULL
);

-- Staff: doctores
CREATE TABLE doctores(
    cedula VARCHAR(10) PRIMARY KEY,
    especialidad VARCHAR(20) NOT NULL,
    FOREIGN KEY (cedula) REFERENCES personas(cedula) ON
    DELETE CASCADE
);

-- Staff: enfermeros
CREATE TABLE enfermeros(
    cedula VARCHAR(10) PRIMARY KEY,
    area VARCHAR(30) NOT NULL,
    FOREIGN KEY (cedula) REFERENCES personas(cedula) ON
    DELETE CASCADE
);

-- Staff: administradores
CREATE TABLE administradores (
    cedula VARCHAR(20) PRIMARY KEY,
    nivel_acceso ENUM('alto', 'medio', 'bajo') DEFAULT
    'medio',
    puede_crear_usuarios BOOLEAN DEFAULT TRUE,
    puede_eliminar_datos BOOLEAN DEFAULT FALSE,
    ultimo_acceso DATETIME,
    FOREIGN KEY (cedula) REFERENCES personas(cedula) ON
    DELETE CASCADE
);

-- Clientes
CREATE TABLE clientes (
    cedula VARCHAR(20) PRIMARY KEY,
```

```

        puede_agendar_citas BOOLEAN DEFAULT TRUE,
        FOREIGN KEY (cedula) REFERENCES personas(cedula) ON
DELETE CASCADE
);

-- Tabla de citas
CREATE TABLE citas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cedula_doctor VARCHAR(20) NOT NULL,
    cedula_cliente VARCHAR(20) NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    departamento VARCHAR(50) NOT NULL,
    estado ENUM('pendiente', 'en_curso', 'completada',
'cancelada') DEFAULT 'pendiente',
    descripcion TEXT NOT NULL,
    observaciones TEXT NOT NULL,
    diagnostico TEXT NOT NULL,
    FOREIGN KEY (cedula_doctor) REFERENCES
doctores(cedula),
    FOREIGN KEY (cedula_cliente) REFERENCES
clientes(cedula),
    CONSTRAINT uc_cita_doctor_hora UNIQUE (cedula_doctor,
fecha, hora)
);

-- Tabla de tratamientos / recetas
CREATE TABLE tratamientos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cita_id INT NOT NULL,
    cedula_doctor VARCHAR(20) NOT NULL,
    cedula_cliente VARCHAR(20) NOT NULL,
    receta TEXT NOT NULL,
    fecha DATETIME DEFAULT NOW(),
    FOREIGN KEY (cita_id) REFERENCES citas(id) ON DELETE
CASCADE,
    FOREIGN KEY (cedula_doctor) REFERENCES
doctores(cedula),
    FOREIGN KEY (cedula_cliente) REFERENCES
clientes(cedula)
);

-- ÍNDICES IMPLEMENTADOS

-- En personas
CREATE UNIQUE INDEX idx_usuario ON personas(usuario);
CREATE UNIQUE INDEX idx_personas_correo ON
personas(correo);
-- para acelerar consultas por rol

```

```

CREATE INDEX idx_personas_rol ON personas(rol);

-- En doctores
CREATE INDEX idx_especialidad ON doctores(especialidad);

-- En enfermeros
CREATE INDEX idx_area ON enfermeros(area);

-- En citas
CREATE INDEX idx_citas_cliente ON citas(cedula_cliente);
CREATE INDEX idx_citas_estado ON citas(estado);
CREATE INDEX idx_citas_fecha ON citas(fecha);

-- En tratamientos
CREATE INDEX idx_tratamientos_doctor ON
tratamientos(cedula_doctor);
CREATE INDEX idx_tratamientos_cliente ON
tratamientos(cedula_cliente);
CREATE INDEX idx_tratamientos_cita ON
tratamientos(cita_id);

```

Justificación: Los índices están estratégicamente diseñados para optimizar las operaciones más críticas del sistema ClinicData. Se enfocan en tres áreas clave: autenticación y control de acceso (índices por usuario y correo), gestión de personal clínico (índices por rol, área y especialidad), y administración de citas médicas (índices por cliente, estado y fecha).

Esta implementación garantiza tiempos de respuesta óptimos en los procesos de mayor impacto diario: acceso al sistema, búsqueda de personal médico especializado, y gestión de citas que representa el núcleo operativo de la clínica. Los índices mejoran significativamente la escalabilidad del sistema, manteniendo un rendimiento consistente conforme crezca el volumen de datos.

6. Implementación Funcional (Frontend)

- **Tecnología:** Java FX, Mysql
- **Funcionalidades finalizadas:** Registro de usuarios, restablecimiento de contraseñas, agendar citas.

- **Operaciones CRUD comprobadas mediante SELECT pos-acción (ver video).

7. Video del Front-end funcional

Link provisional:

<https://drive.google.com/file/d/1ofRnyuJRiRvUumrqd9H56wrubBb8hBta/view?usp=sharing>

(duración 4 min con 39 s).

8. Procedimientos Almacenados

```
-- procedimientos almacenados (pendientes todavía)
-- registrar usuarios (admins)

DELIMITER //

CREATE PROCEDURE sp_registrar_usuario(
    IN p_cedula VARCHAR(10),
    IN p_nombre VARCHAR(15),
    IN p_apellido VARCHAR(15),
    IN p_edad INT,
    IN p_correo VARCHAR(30),
    IN p_telefono VARCHAR(10),
    IN p_direccion VARCHAR(20),
    IN p_usuario VARCHAR(10),
    IN p_contrasena VARCHAR(255),
    IN p_rol ENUM('admin', 'staff', 'enfermero', 'cliente'),
    IN p_especialidad VARCHAR(20) -- Solo para doctores si
aplica
)
BEGIN
    START TRANSACTION;
    -- Insertar en tabla personas
    INSERT INTO personas (cedula, nombre, apellido, edad,
correo, telefono, direccion1, usuario, contraseña, rol)
        VALUES (p_cedula, p_nombre, p_apellido, p_edad,
p_correo, p_telefono, p_direccion, p_usuario, p_contrasena,
p_rol);

```

```

-- Insertar en tabla específica según rol
-- doctores

IF p_rol = 'staff' AND p_especialidad IS NOT NULL THEN
    INSERT INTO doctores (cedula, especialidad) VALUES
(p_cedula, p_especialidad);

-- admins

ELSEIF p_rol = 'admin' THEN
    INSERT INTO administradores (cedula, nivel_acceso)
VALUES (p_cedula, 'medio');

END IF;

COMMIT;

SELECT 'Registro exitoso' AS mensaje;
END //
DELIMITER ;

```

- Bloqueo de filas por PK, control transaccional, manejo de errores.
- 4 procedimientos adicionales en desarrollo (60 % completado).faltan 3

9. Disparadores (Triggers)

```

-- triggers
-- validación de fechas de citas, cambiar su estado a vencida si
el cliente no llega a tiempo
DELIMITER //
create trigger tgr_actualizar_fecha_citas
before insert on citas
for each row
begin
    if new.fecha < curdate() then
        set new.estado = 'vencida';
    end if;
end///
DELIMITER ;

-- mostrar triggers
show triggers from clinicdb;

```

10. Consultas SQL y Vistas

```

5     VIEW `clinicdb`.`new_view` AS
6         SELECT
7             `p`.`cedula` AS `cedula`,
8             `p`.`nombre` AS `nombre`,
9             `p`.`apellido` AS `apellido`,
10            `p`.`edad` AS `edad`,
11            `p`.`correo` AS `correo`,
12            `p`.`telefono` AS `telefono`,
13            `p`.`direccion1` AS `direccion1`,
14            `p`.`usuario` AS `usuario`,
15            `p`.`rol` AS `rol`,
16            (CASE
17                WHEN (`p`.`rol` = 'admin') THEN `a`.`nivel_acceso`
18                WHEN (`p`.`rol` = 'staff') THEN `d`.`especialidad`
19                WHEN (`p`.`rol` = 'enfermero') THEN `e`.`area`
20                WHEN (`p`.`rol` = 'cliente') THEN 'cliente'
21            END) AS `detalle_por_rol`,
22            (CASE
23                WHEN (`p`.`rol` = 'admin') THEN `a`.`ultimo_acceso`
24                ELSE NULL
25            END) AS `ultimo_acceso`
26        FROM
27            (((`clinicdb`.`personas` `p`
28            LEFT JOIN `clinicdb`.`administradores` `a` ON ((`p`.`cedula` = `a`.`cedula`)))
29            LEFT JOIN `clinicdb`.`doctores` `d` ON ((`p`.`cedula` = `d`.`cedula`)))
30            LEFT JOIN `clinicdb`.`enfermeros` `e` ON ((`p`.`cedula` = `e`.`cedula`)))
31

```

1. Listar todos los usuarios del sistema con sus roles

```
SELECT cedula, nombre, apellido, rol FROM personas ORDER BY rol,
apellido;
```

2. Cantidad de personas por rol

```
SELECT rol, COUNT(*) as cantidad
FROM personas
GROUP BY rol
ORDER BY cantidad DESC;
```

11. Organización del Proyecto:

```

gestion_clinica/
└── .idea/
└── .mvn/
└── src/
    └── main/

```

```
java/
└── com.espol.proyectodb3/
    ├── AdminController
    ├── ClientController
    ├── DoctorController
    ├── EliminarUsuarioController
    ├── MainApp
    ├── MainController
    ├── NuevaCitaController
    ├── RecoverPasswordViewController
    ├── RegisterViewController
    ├── RegistrarStaffController
    └── entidades/
        ├── Admin
        ├── Cita
        ├── Cliente
        ├── Doctor
        ├── Persona
        ├── Recetas
        └── Tratamiento
    └── seguridad_Otros/
        └── Utilleria
    └── SQL/
module-info.java
resources/
└── com.espol.proyectodb3/
    ├── Admin-view.fxml
    ├── Client-view.fxml
    ├── Doctor-view.fxml
    ├── EliminarUsuarioView.fxml
    ├── Main-view.fxml
    ├── NuevaCita-view.fxml
    ├── recoverPassword-view.fxml
    ├── Register-view.fxml
    └── RegistrarStaff-view.fxml
```

