



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



PROGRAMACION ORIENTADAS A OBJETOS (TDS314) GR1

PROFESOR: YADIRA FRANCO

FECHA: 30 de julio de 2025

PERÍODO ACADÉMICO: 2025-A

PROYECTO FINAL DE PROGRAMACIÓN ORIENTADA A OBJETOS

Nombre:

Bryan Andrés Angulo Sarango

Link de Git

<https://github.com/BryanAngulo1022/proyectoSiSalud.git>

Índice

Contenido

1.	Título del Proyecto	3
2.	Objetivos	3
a.	Objetivo General.....	3
b.	Objetivos Específicos	3
3.	Justificación	3
4.	Alcance y Requerimientos – roles.....	4
a.	Requerimientos funcionales	4
b.	Requerimientos Tecnológicos	6
5.	Diseño de Interfaz (Mockup)	7
a.	Formulario de ingreso (Login).....	7
b.	Menú de administración	7
c.	Gestión de doctores	8
d.	Gestión de pacientes.....	8
e.	Menú de recepción	9
6.	Modelado de Base de Datos.....	9
a.	Entidades y Relaciones	9
b.	Diagrama Entidad Relación	10
7.	Organización del Código	11
8.	Ejecución Completa del Proyecto	13
9.	Explicación del Desarrollo	21
a.	Vistas: Interfaces gráficas, eventos.....	21
b.	Conector.....	21
c.	Conexión	21
d.	Main.....	23
e.	Ejecutable.....	23
10.	Repositorio GitHub y .exe(ejecutable)	25
11.	Funcionamiento con Base de Datos en la Nube.....	25

12.	Conclusiones	26
13.	Recomendaciones.....	26
14.	Anexos	27

Índice de Ilustraciones

Ilustración 1. Formulario de ingreso	7
Ilustración 2. Formulario de menú de administración	7
Ilustración 3. Formulario de gestión de doctores.....	8
Ilustración 4. Formulario de gestión de pacientes	8
Ilustración 5. Menú de recepción	9
Ilustración 6. Diagrama Entidad Relación	10
Ilustración 7. Inicio de sesión del proyecto.....	13
Ilustración 8. Menú en rol administrador por pestañas.....	14
Ilustración 9.Menu rol recepcionista por pestañas.....	15
Ilustración 10. Ejemplo registro de pacientes.....	15
Ilustración 11. Ejemplo actualización de pacientes	16
Ilustración 12. Verificación de la creación y actualización en la BD.....	16
Ilustración 13. Ejemplo de eliminar pacientes.	16
Ilustración 14. Verificación de eliminar paciente en la BD.....	17
Ilustración 15. Ejemplo de lectura de pacientes.	17
Ilustración 16. Datos iniciales en la tabla doctores de la BD.....	17
Ilustración 17. Ejemplo del registro de doctores.....	18
Ilustración 18. Verificación de la creación de doctores en la BD.....	18
Ilustración 19. Ejemplo actualización de doctores.....	18
Ilustración 20. Verificación de la actualización de doctores en la BD.....	19
Ilustración 21. Ejemplo de lectura de doctores.....	19
Ilustración 22. Entorno de PHPMyAdmin con la tabla de doctores.....	20
Ilustración 23. Entorno de PHPMyAdmin con la tabla de pacientes.....	20
Ilustración 24. Credenciales proporcionadas por Clever Cloud para la conexión con MySQL.	22
Ilustración 25. Credenciales (URI) proporcionadas por Clever Cloud para la conexión con MySQL	22
Ilustración 26. Conexión en DBeaver	23
Ilustración 27. Entorno de Launch4j y configuración del ejecutable	24
Ilustración 28. Entorno de Launch4j con la carga del JDK 24	24
Ilustración 29. Script del proyecto para la creación de la base de datos.....	27
Ilustración 30. Script para cargar datos iniciales	28

1. Título del Proyecto

Proyecto 2: SISALUD - Sistema de Gestión de Citas Médicas

2. Objetivos

a. Objetivo General

Desarrollar un sistema de escritorio utilizando Java aplicando los principios de Programación Orientada a Objetos (POO), que permita el registro de pacientes, agendar y eliminar citas, ver el historial de citas agendadas, registrar doctores y generar reportes por especialidad. El sistema debe tener acceso seguro y diferenciado por roles (Administrador y Recepcionista), validaciones obligatorias y navegación intuitiva, con conexión a base de datos en la nube.

b. Objetivos Específicos

- Implementar los principios de Programación Orientada a Objetos (POO).
- Diseñar una interfaz gráfica profesional con Java Swing.
- Organizar el código en paquetes con estructura clara y mantenible.
- Conectar el sistema a una base de datos remota MySQL.
- Generar reportes y controlar registros con validaciones de los campos.

3. Justificación

El presente proyecto tiene como finalidad fortalecer los conocimientos adquiridos en Programación Orientada a Objetos (POO) mediante el desarrollo de un caso práctico que emula el funcionamiento de un sistema real. A través de esta simulación, se busca que como estudiantes apliquemos de forma efectiva los pilares fundamentales de la POO, integrándolos en una solución funcional y coherente con escenarios del mundo real.

Este enfoque práctico permite no solo afianzar la teoría aprendida, sino también comprender la importancia del diseño estructurado en el desarrollo de software. La implementación del sistema se basa en principios de diseño limpio, promoviendo un código claro, organizado y fácil de mantener.

4. Alcance y Requerimientos – roles

a. Requerimientos funcionales

1. Login (acceso al sistema)

- El sistema permite el inicio de sesión con usuario y contraseña diferenciado por rol.
- El sistema permite seleccionar los roles: Administrador y Recepcionista.
- Cada rol tendrá acceso a funcionalidades específicas.

Requerimiento en Rol Administrador

1. Registro de pacientes (gestión de pacientes)

- El sistema permite registrar, editar, eliminar y consultar pacientes CRUD.
- Datos requeridos: nombre, cédula, fecha de nacimiento, teléfono, dirección, género y correo.
- Validaciones: fecha de nacimiento, cédula única, correo y teléfono válido.

2. Registro de doctores (gestión de doctores)

- El sistema permite registrar, editar, eliminar y consultar doctores.
- Datos requeridos: nombre, cédula, especialidad, jornada, correo y teléfono.
- Validaciones: cédula única, correo y teléfono válido.

3. Citas

- El sistema permite buscar, ver y eliminar las citas agendadas por paciente.
- Datos requeridos: cédula
- Validaciones: campos vacíos

4. Reportes

- El sistema permite generar un reporte de citas atendidas y por atender, diferenciado por especialidad y por doctor.
- El sistema genera un reporte general del número total de citas atendidas y por atender.

Requerimiento en Rol Recepcionista

1. Registro de pacientes (gestión de pacientes)

- El sistema permite registrar pacientes.

- Datos requeridos: nombre, cédula, fecha de nacimiento, teléfono, dirección, género y correo.
- Validaciones: fecha de nacimiento, cédula única, correo y teléfono válido.

2. Agendamiento de citas

- El sistema permite crear citas médicas.
- Las citas se asignan seleccionando la especialidad, el nombre del doctor, el día de la cita a partir del día siguiente al que se está atendiendo y el horario disponible en intervalos de media hora.
- Datos requeridos: cédula
- Validaciones: campos vacíos

3. Historial médico

- El sistema permite visualizar el historial de las citas agendadas por el paciente.
- Datos requeridos: cédula
- Validaciones: campos vacíos

b. Requisitos No Funcionales

- **Diseño responsivo en ventanas**

Las interfaces desarrolladas en Java Swing se adaptan correctamente a diferentes tamaños de pantalla, facilitando el uso tanto en monitores estándar como en resoluciones más amplias.

- **Conección a base de datos remota**

El sistema se conecta a una base de datos My SQL alojada en la nube de Clever Cloud. Esto permite disponibilidad permanente, acceso remoto al sistema y facilita la gestión centralizada de la información clínica.

- **Código limpio y comentado**

La implementación del sistema sigue buenas prácticas de programación orientada a objetos. El código está estructurado y comentado, explicando clases, métodos y funcionalidades clave, lo que facilita su mantenimiento y futuras ampliaciones.

- **Separación por paquetes**

La estructura del proyecto está organizada en paquetes específicos como Main, Conexion, Roles (Repcionista y Administrador), AdministradorCRUD, Imágenes y Validaciones, lo que favorece una arquitectura modular, clara y fácil de entender.

b. Requerimientos Tecnológicos

- **Java JDK 21**

El proyecto fue desarrollado usando Java JDK 24, compatible con la sintaxis moderna de Java y nuevas características del lenguaje.

- **IDE IntelliJ IDEA**

Todo el desarrollo se hizo en IntelliJ IDEA, aprovechando sus herramientas de diseño GUI.

- **Base de Datos MySQL en nube**

El sistema está conectado a una base de datos MySQL alojada en Clever Cloud, accesible remotamente a través del jdbc:mysql://... con las credenciales del Clever Cloud.

- **Librerías JDBC**

El acceso a la base de datos se hace mediante JDBC (mysql-connector-j-9.3.0), utilizando java.sql.Connection, PreparedStatement, ResultSet, etc., con manejo de errores y consultas SQL seguras.

- **GitHub para control de versiones**

El proyecto se encuentra alojado en un repositorio en GitHub, con los commits correspondientes que muestran la evolución y avance del proyecto.

5. Diseño de Interfaz (Mockup)

Para garantizar una experiencia de usuario intuitiva y ordenada, se realizó el diseño preliminar de las interfaces del sistema **SISALUD** utilizando la herramienta **NinjaMock**. Esta herramienta permitió representar de forma visual y estructurada la distribución de los componentes de cada formulario antes de implementar el código en Java Swing.

a. Formulario de ingreso (Login)

The form is titled "LOGIN". It contains three input fields: "Usuario" with placeholder "ingresar usuario", "Clave" with placeholder "ingresar clave", and "Rol" with placeholder "seleccionar rol" and a dropdown arrow icon. Below the fields is a "Acceder" button.

Ilustración 1. Formulario de ingreso

b. Menú de administración

The mockup shows three panels of the administration menu:

- Panel 1 (Main Menu):** Titled "Administrador". It has a "Bienvenid@, Nombre" label, a "Cerrar Sesión" button, and three buttons: "Gest. Usuarios", "Citas", and "Reporte". Below is a section "Ir a gestión de:" with icons for "Pacientes" and "Doctores".
- Panel 2 (Citas Management):** Titled "Administrador". It has the same header and buttons. Below is a "Buscar Citas" section with a "Cedula" input field, "Limpiar" and "Buscar" buttons, and a "Lista de citas pendientes:" text area containing three radio buttons: "Option 1", "Option 2" (selected), and "Option 3".
- Panel 3 (Report Generation):** Titled "Administrador". It has the same header and buttons. Below is a "Reportes por especialidad" section with an "Especialidad" dropdown, a "Ver Reporte" button, and a table with three columns: "Column 1", "Column 2", and "Column 3". Below is a "Reporte General" section with a "Ver Reporte" button and a table with three columns: "Column 1", "Column 2", and "Column 3".

Ilustración 2. Formulario de menú de administración

c. Gestión de doctores

The image displays three wireframe screens for managing doctors:

- Screen 1: Creación de nuevos doctores**
 - Buttons: Registro, Actualizar o eliminar, Ver.
 - Form fields:
 - Nombre: Text input
 - Cédula: Text input
 - Especialidad: Text input
 - Jornada: Text input
 - Correo: Text input
 - Teléfono: Text input
 - Buttons: Guardar, Regresar
- Screen 2: Gestión de doctores**
 - Buttons: Registro, Actualizar o eliminar, Ver.
 - Form fields:
 - Buscar doctor: Cédula: Text input, Buscar cargar
 - Actualizar Doctores: Nombre: Text input, Especialidad: Text input, Jornada: Text input, Correo: Text input, Teléfono: Text input
 - Buttons: Actualizar, Eliminar, Regresar
- Screen 3: Gestión de doctores**
 - Buttons: Registro, Actualizar o eliminar, Ver.
 - Form fields:
 - Mostrar Doctores Registrados: Text area
 - Buttons: Regresar

Ilustración 3. Formulario de gestión de doctores

d. Gestión de pacientes

The image displays three wireframe screens for managing patients:

- Screen 1: Creación de nuevos pacientes**
 - Buttons: Registro, Actualizar o eliminar, Ver.
 - Form fields:
 - Nombre: Text input
 - Cédula: Text input
 - Fecha de nacim: Text input
 - Teléfono: Text input
 - Dirección: Text input
 - Género: Text input
 - Correo: Text input
 - Buttons: Guardar, Regresar
- Screen 2: Gestión de pacientes**
 - Buttons: Registro, Actualizar o eliminar, Ver.
 - Form fields:
 - Buscar paciente: Cédula: Text input, Buscar cargar
 - Actualizar Pacientes: Nombre: Text input, Fecha de nacim: Text input, Teléfono: Text input, Dirección: Text input, Género: Text input, Correo: Text input
 - Buttons: Actualizar, Eliminar, Regresar
- Screen 3: Gestión de pacientes**
 - Buttons: Registro, Actualizar o eliminar, Ver.
 - Form fields:
 - Mostrar Pacientes Registrados: Text area
 - Buttons: Regresar

Ilustración 4. Formulario de gestión de pacientes

e. Menú de recepción

The figure consists of three separate wireframe diagrams of a computer screen, each representing a different page of a receptionist's menu. All three screens have a header bar with the title 'Recepcionista' and a 'Cerrar Sesión' button in the top right corner. Below the header, there are three buttons: 'Reg. Paciente', 'Agendar Cita', and 'Hist. Med.'. The first screen shows a 'Registro de Pacientes' section with fields for Nombre, Cédula, Fecha de nacimiento, Teléfono, Genero, and Correo, each with a 'Text input' placeholder. A 'Guardar' button is at the bottom. The second screen shows a 'Buscar Paciente:' section with a 'Cedula' field and a 'Buscar' button, followed by a 'Registrar cita' section with dropdown menus for Especialidad, Doctor/a, Dia de la cita, and Hora de la cita, and an 'Agendar cita' button. The third screen shows a 'Historial médico' section with a 'Cedula' field, 'Limpiar' and 'Buscar' buttons, and a large 'Text area' for medical history notes.

Ilustración 5. Menú de recepción

6. Modelado de Base de Datos

a. Entidades y Relaciones

USUARIO

- id_usuario (PK)
- nombre
- usuario (único)
- correo (único)
- contraseña
- rol: *Administrador o Recepcionista*

PACIENTE

- id_paciente (PK)
- nombre
- ci (único)
- fecha_nacimiento
- teléfono
- dirección
- genero
- correo

ESPECIALIDAD

- id_especialidad (PK)
- nombre

DOCTOR

- id_doctor (PK)
- nombre
- ci (único)
- id_especialidad (FK) → ESPECIALIDAD
- correo
- teléfono
- jornada

CITA

- id_cita (PK)
- id_paciente (FK) → PACIENTE
- id_doctor (FK) → DOCTOR
- fecha (único)
- hora (único)

Restricción única: un doctor no puede tener 2 citas en la misma fecha y hora.

b. Diagrama Entidad Relación

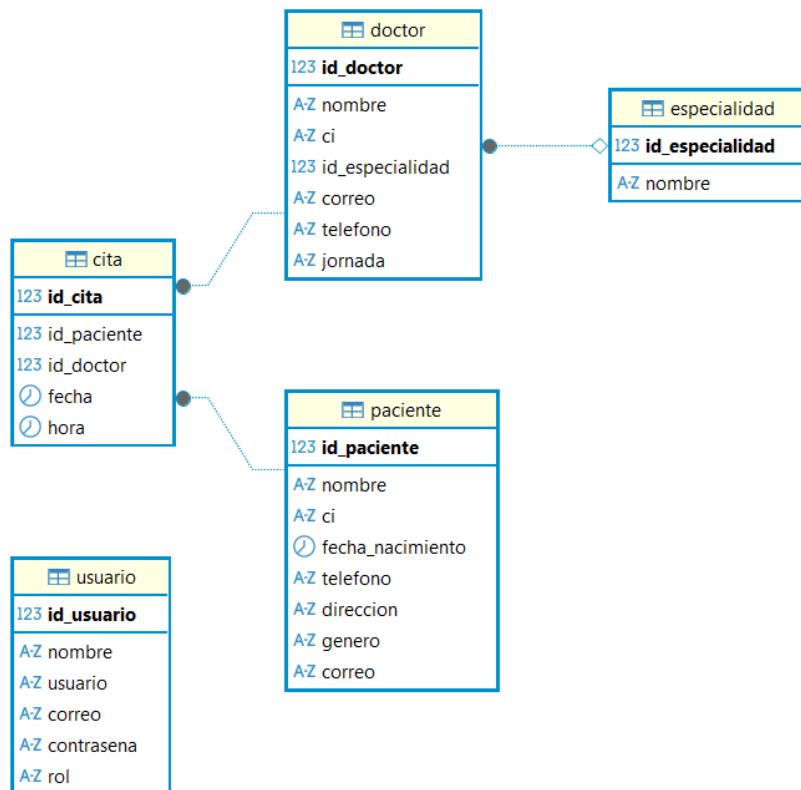
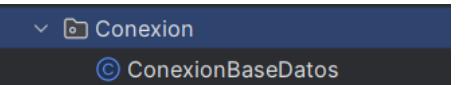
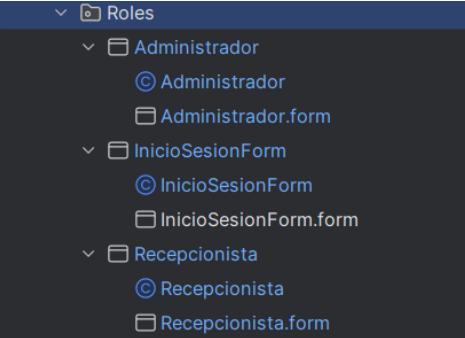
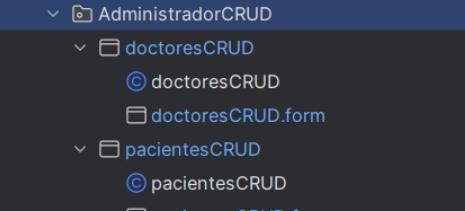
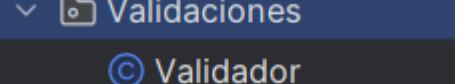
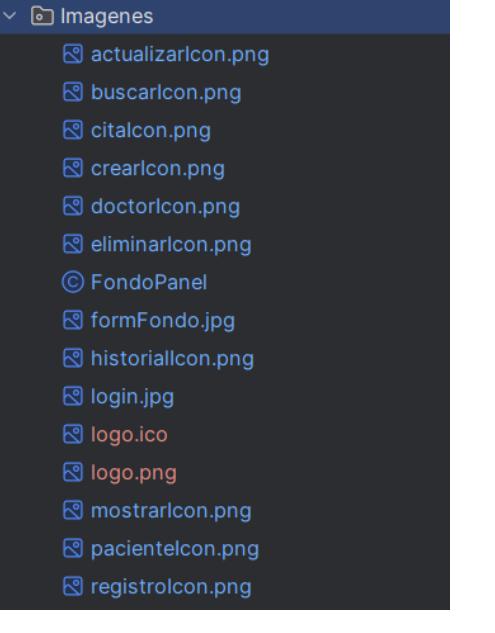


Ilustración 6. Diagrama Entidad Relación

7. Organización del Código

Para la organización del código se crearon paquetes agrupados en función de la característica de los archivos.

Nombre del Paquete	Descripción	Componentes
Main	Está conformado por el ejecutable principal del programa, desde aquí se inicia el programa principal.	
Conexion	Está conformado por la clase de conexión, permite hacer la conexión directa a la base de datos en MySQL.	
Roles	Está conformado por la clase de inicio de sesión y los roles (Administrador y recepcionista) compuestos cada uno por archivos .java y .form que se encarga de la creación de las interfaces (formularios Swing) y de las acciones que cada interfaz realizará diferenciada por los roles.	
AdministradorCRUD	Está conformado por los archivos .java y .form que se encarga de la creación de las interfaces (formularios Swing) y de las acciones propias del CRUD de pacientes y doctores validadas únicamente para el rol administrador.	
Validaciones	Esta confirmado por la clase que permiten ejecutar las validaciones cédula única, correo y teléfono válido.	

imágenes	Está conformado por la imágenes e iconos utilizar en la interfaz visual.	 <p>The screenshot shows a file explorer window with a dark theme. A folder named "Imagenes" is selected, indicated by a blue header bar. Inside the folder, there are several files listed:</p> <ul style="list-style-type: none">actualizarIcon.pngbuscarIcon.pngcitalcon.pngcrearIcon.pngdoctorIcon.pngeliminarIcon.pngFondoPanelformFondo.jpghistorialIcon.pnglogin.jpglogo.icologo.pngmostrarIcon.pngpacientelcon.pngregistroIcon.png
----------	--	---

8. Ejecución Completa del Proyecto

- Inicio del sistema desde la clase Main.

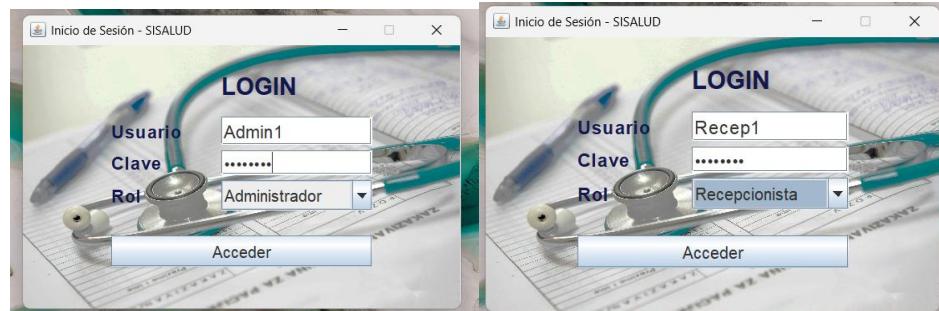
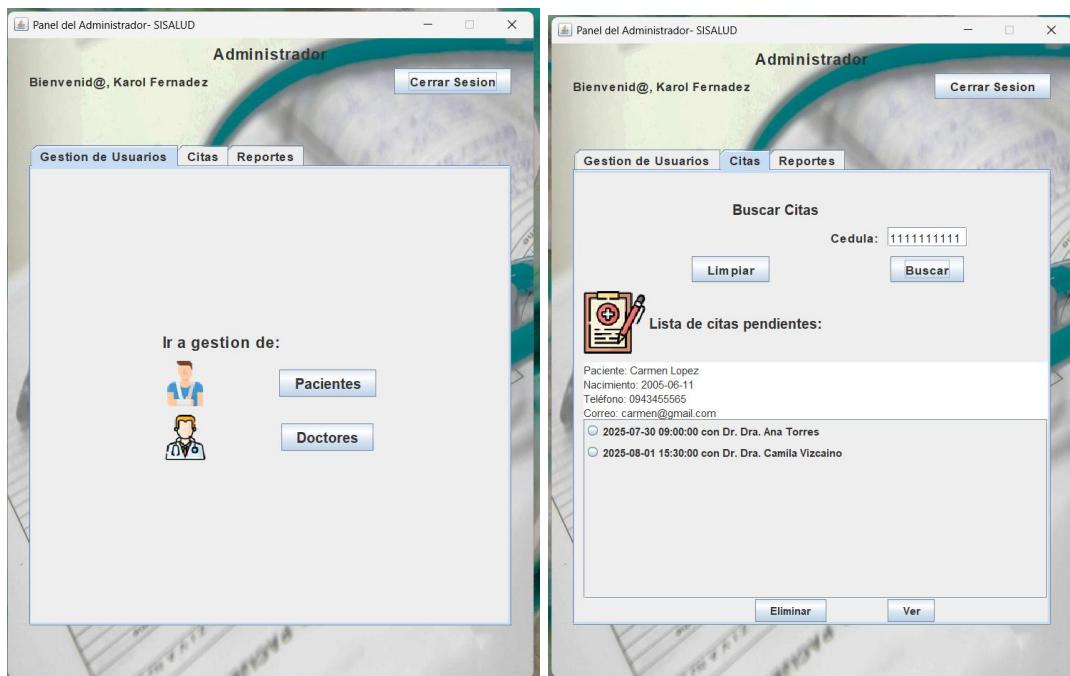


Ilustración 7. Inicio de sesión del proyecto

- Menú principal de navegación.

Menú en rol administrador



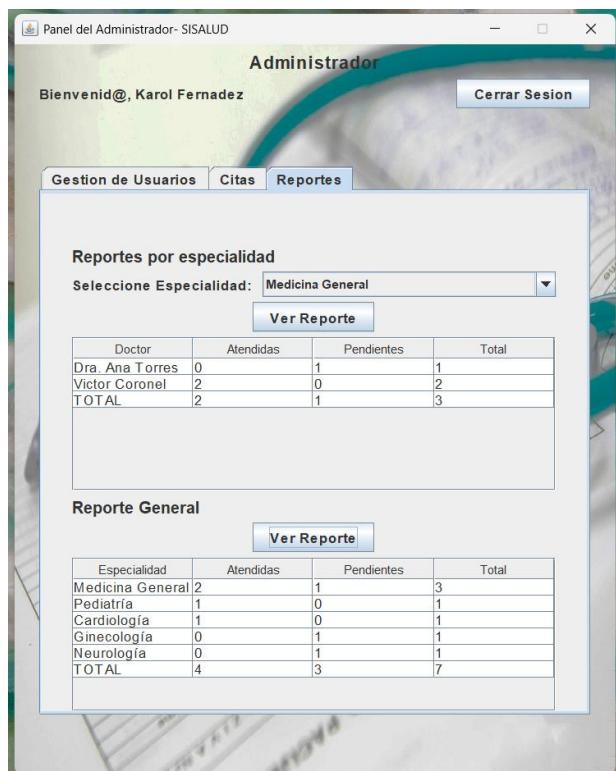


Ilustración 8. Menú en rol administrador por pestañas.

Menu rol recepcionista



Ilustración 9. Menú rol recepcionista por pestañas.

- CRUD completo desde formularios comparada con la conexión en la base de datos.

CRUD de pacientes

Ilustración 10. Ejemplo registro de pacientes.

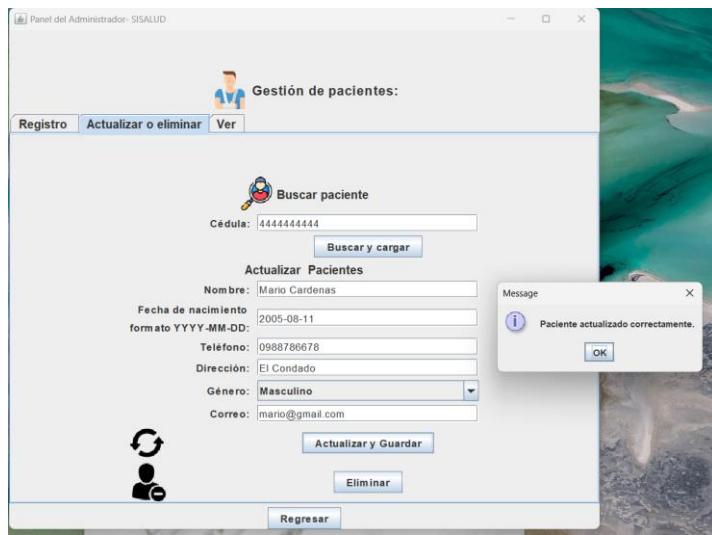


Ilustración 11. Ejemplo actualización de pacientes.

	id_paciente	nombre	ci	fecha_nacimiento	telefono	direccion	genero	correo
1	1	Bryan Angulo	1717839235	2000-06-22	0973884938	Calderon	Masculino	bryan@gmail.com
2	2	Pedro Freire	1234567890	1990-01-22	2342345345	Dammer	Masculino	freire@gmail.com
3	4	Pedro Perez	2222222222	2001-06-20	4343456754	El Capuli	Masculino	pedro@gmail.com
4	5	Luci Madrid	3333333333	2002-12-21	2392034540	Calderon	Masculino	luci@gmail.com
5	7	Patricio Sevilla	1010101033	2000-05-21	242312059	El Quinche	Masculino	particio@gmail.com
6	8	Dante Sevilla	1717828293	2000-04-21	0934333222	Carcelen	Masculino	dante@gmail.com
7	9	Carmen Lopez	1111111111	2005-06-11	0943455565	Centro Historico	Femenino	carmen@gmail.com
8	11	Pedro Gonzales	0400433235	2020-12-31	0933321213	El Porvenir	Masculino	pedrog@gmail.com
9	12	Mario Cardenas	4444444444	2005-08-11	0988786678	El Condado	Masculino	mario@gmail.com
10	13	Alejandra Mendoza	5555555555	2010-05-05	3434555345	El Bosque	Femenino	aleja@gmail.com

Ilustración 12. Verificación de la creación y actualización en la BD.

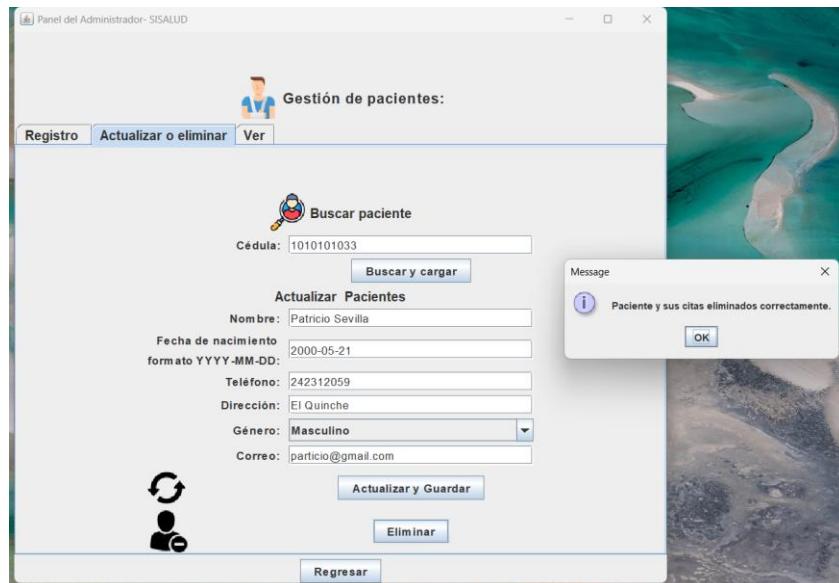


Ilustración 13. Ejemplo de eliminar pacientes.

Propiedades Datos Diagrama

Show SQL Enter a SQL expression to filter results (use Ctrl+Space)

	123 < id_paciente	AZ nombre	AZ ci	AZ fecha_nacimiento	AZ telefono	AZ direccion	AZ genero	AZ correo
Grilla	1	Bryan Angulo	1717839235	2000-06-22	0973884938	Calderon	Masculino	bryan@gmail.com
Texto	2	Pedro Freire	1234567890	1990-01-22	2342345345	Dammer	Masculino	freire@gmail.com
4	3	Pedro Perez	2222222222	2001-06-20	4343456754	El Capuli	Masculino	pedro@gmail.com
5	4	Luci Madrid	3333333333	2002-12-21	2392034540	Calderon	Masculino	luci@gmail.com
8	5	Dante Perez	1717828293	2000-04-21	0934333222	Carcelen	Masculino	dante@gmail.com
9	6	Carmen Lopez	1111111111	2005-06-11	0943455565	Centro Historico	Femenino	carmen@gmail.com
11	7	Pedro Gonzales	0400433235	2020-12-31	0933321213	El Porvenir	Masculino	pedrog@gmail.com
12	8	Mario Cardenas	4444444444	2005-08-11	0988786678	El Condado	Masculino	mario@gmail.com
13	9	Alejandra Mendoza	5555555555	2010-05-05	3434555345	El Bosque	Femenino	aleja@gmail.com

Ilustración 14. Verificación de eliminar paciente en la BD.

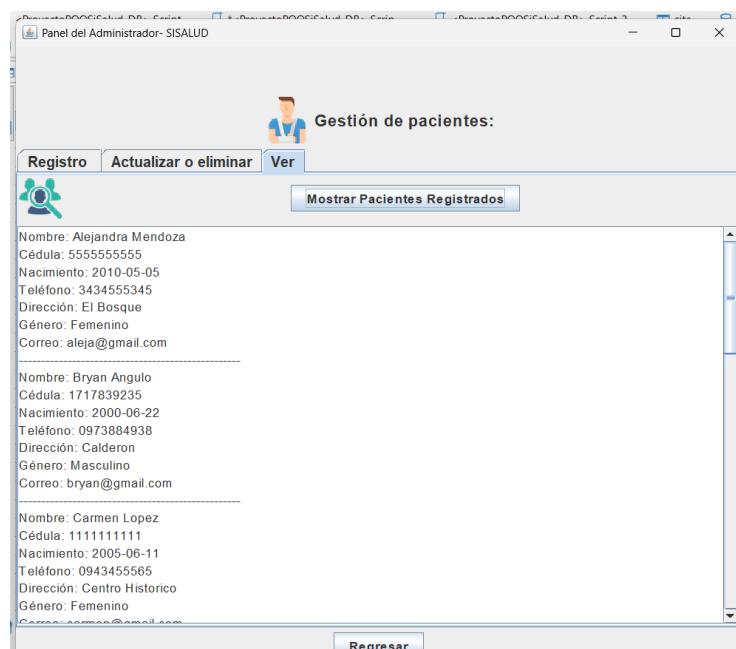


Ilustración 15. Ejemplo de lectura de pacientes.

CRUD doctores

Propiedades Datos Diagrama

Show SQL Enter a SQL expression to filter results (use Ctrl+Space)

	123 < id_doctor	AZ nombre	AZ ci	123 < id_especialidad	AZ correo	AZ telefono	AZ jornada
Grilla	1	Dra. Ana Torres	1234312349	1	ana.torres@hospital.com	0987654321	Mañana
Texto	2	Dr. Juan Pérez	4939857477	2	juan.perez@hospital.com	0987456123	Tarde
3	3	Dr. Carlos Díaz	5431232349	3	carlos.diaz@hospital.com	0991234567	Completa
4	4	Dra. Camila Vizcaino	1093123249	4	cam.viz@hospital.com	0983466123	Tarde
5	5	Dra. Ana Torres	0312323490	5	ana.torres@hospital.com	0967845679	Completa
6	6	Victor Coronel	0496588590	1	coronel@hospital.com	2567893245	Tarde
7	7	Manuel Ortiz	2121212100	3	manuel@hospital.com	0948492200	Completa

Ilustración 16. Datos iniciales en la tabla doctores de la BD.

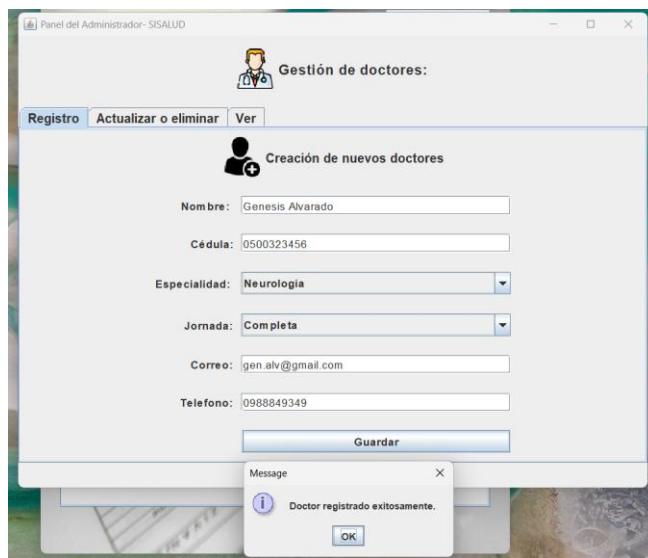


Ilustración 17. Ejemplo del registro de doctores.

	id_doctor	nombre	ci	id_especialidad	correo	telefono	jornada
1	1	Dra. Ana Torres	123412349	1	ana.torres@hospital.com	0987654321	Mañana
2	2	Dr. Juan Pérez	4939857477	2	juan.perez@hospital.com	0987456123	Tarde
3	3	Dr. Carlos Díaz	5431232349	3	carlos.diaz@hospital.com	0991234567	Completa
4	4	Dra. Camila Vizcaino	1093123249	4	cam.viz@hospital.com	0983466123	Tarde
5	5	Dra. Ana Torres	0312323490	5	ana.torres@hospital.com	0967845679	Completa
6	6	Victor Coronel	0496588590	1	coronel@hospital.com	2567893245	Tarde
7	7	Manuel Ortiz	2121212100	3	manuel@hospital.com	0948492200	Completa
8	8	Genesis Alvarado	0500323456	5	gen.alv@gmail.com	0988849349	Completa

Ilustración 18. Verificación de la creación de doctores en la BD.

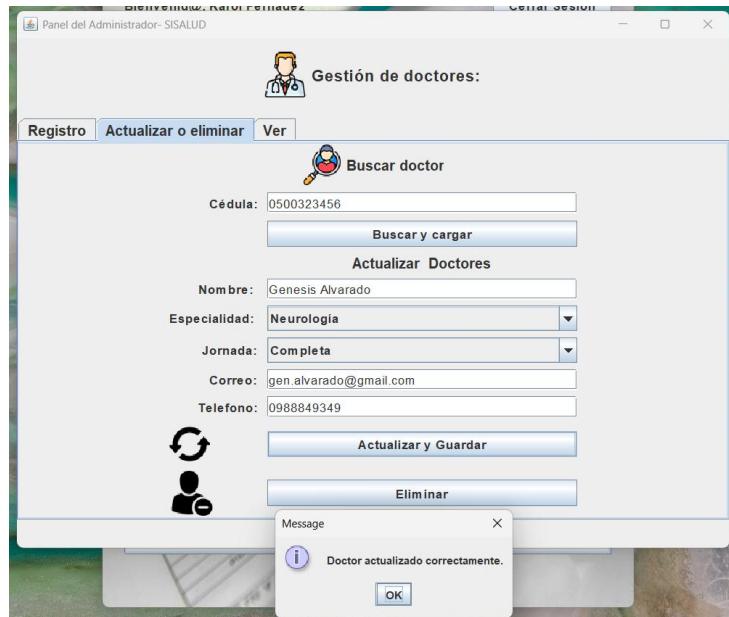


Ilustración 19. Ejemplo actualización de doctores.

	123 id_doctor	AZ nombre	AZ ci	123 id_especialidad	AZ correo	AZ telefono	AZ jornada
1	1	Dra. Ana Torres	1234312349	1	ana.torres@hospital.com	0987654321	Mañana
2	2	Dr. Juan Pérez	4939857477	2	juan.perez@hospital.com	0987456123	Tarde
3	3	Dr. Carlos Díaz	5431232349	3	carlos.diaz@hospital.com	0991234567	Completa
4	4	Dra. Camila Vizcaino	1093123249	4	cam.viz@hospital.com	0983466123	Tarde
5	5	Dra. Ana Torres	0312323490	5	ana.torres@hospital.com	0967845679	Completa
6	6	Victor Coronel	0496588590	1	coronel@hospital.com	2567893245	Tarde
7	7	Manuel Ortiz	2121212100	3	manuel@hospital.com	0948492200	Completa
8	8	Genesis Alvarado	0500323456	5	gen.alvarado@gmail.com	0988849349	Completa

Ilustración 20. Verificación de la actualización de doctores en la BD.

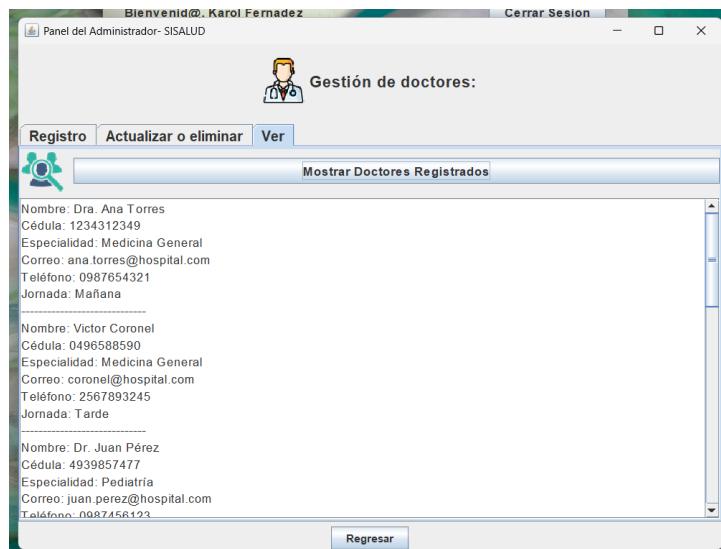


Ilustración 21. Ejemplo de lectura de doctores.

- Conexión exitosa con MySQL en la nube.

Se presenta el entorno PHPMyAdmin, alojado en Clever Cloud, con las tablas y registros creados en la base de datos MySQL.

The screenshot shows the MySQL by Clever Cloud dashboard on the left and the PHPMyAdmin interface on the right. The dashboard lists various add-ons like 'ProductosCuidadoPers...' and 'ProyectoPOOSiSalud'. The selected 'Doc. for MySQL' add-on is expanded, showing its schema structure under 'Cita' (Columns and Indices) and other tables like 'doctor', 'especialidad', 'paciente', and 'usuario'. The main area displays the 'doctores' table with the following data:

	id_doctor	nombre	ci	id_especialidad	correo	telefono	jornada
1	Dra. Ana Torres	1234312349	1	ana.torres@hospital.com	0987654321	Mañana	
2	Dr. Juan Pérez	4939857477	2	juan.perez@hospital.com	0987456123	Tarde	
3	Dr. Carlos Díaz	5431232349	3	carlos.diaz@hospital.com	0991234567	Completa	
4	Dra. Camila Vizcaino	1093123249	4	cam.viz@hospital.com	0983466123	Tarde	
5	Dr. Ana Torres	0312323490	5	ana.torres@hospital.com	0967845679	Completa	
6	Victor Coronel	0496588590	1	coronel@hospital.com	2567893245	Tarde	
7	Manuel Ortiz	2121212100	3	manuel@hospital.com	0948492200	Completa	
8	Genesis Alvarado	0500323456	5	gen.alvarado@gmail.com	0988849349	Completa	

Ilustración 22. Entorno de PHPMyAdmin con la tabla de doctores.

The screenshot shows the MySQL by Clever Cloud dashboard on the left and the PHPMyAdmin interface on the right. The selected 'Doc. for MySQL' add-on is expanded, showing its schema structure under 'Cita' (Columns and Indices) and other tables like 'doctor', 'especialidad', 'paciente', and 'usuario'. The main area displays the 'pacientes' table with the following data:

	id_paciente	nombre	ci	fecha_nacimiento	telefono	direccion	genero	correo
1	Bryan Angulo	1717839235	2000-06-22	0973884938	Calderon	Masculino	bryan@gmail.com	
2	Pedro Freire	1234567890	1990-01-22	2342345345	Dammer	Masculino	freire@gmail.com	
4	Pedro Perez	2222222222	2001-06-20	4343456754	El Capuli	Masculino	pedro@gmail.com	
5	Luci Madrid	3333333333	2002-12-21	2392034540	Calderon	Masculino	luci@gmail.com	
8	Dante Perez	1717828293	2000-04-21	0934333222	Carcelen	Masculino	dante@gmail.com	
9	Carmen Lopez	1111111111	2005-06-11	0943455565	Centro Historico	Femenino	carmen@gmail.com	
11	Pedro Gonzales	0400433235	2020-12-31	0933321213	El Porvenir	Masculino	pedrog@gmail.com	
12	Mario Cardenas	4444444444	2005-08-11	0988786678	El Condado	Masculino	mario@gmail.com	
13	Alejandra Mendoza	5555555555	2010-05-05	3434555345	El Bosque	Femenino	aleja@gmail.com	

Ilustración 23. Entorno de PHPMyAdmin con la tabla de pacientes.

9. Explicación del Desarrollo

El desarrollo del sistema **SISALUD** se basó en una arquitectura modular y estructurada, siguiendo los principios de separación de responsabilidades y buenas prácticas de programación orientada a objetos. A continuación, se detallan los principales componentes desarrollados:

a. Vistas: Interfaces gráficas, eventos.

Las interfaces gráficas de usuario fueron implementadas utilizando **Java Swing**, y se organizaron según los distintos roles del sistema. Cada formulario cumple una función específica dentro del sistema:

- **LoginForm**: Validación de acceso al sistema.
- **Administrador**: Acceso a formularios de gestión (usuarios, doctores, pacientes).
- **Repcionista**: Registro de pacientes y agendamiento de citas.
- **PacientesCRUD, DoctoresCRUD, UsuarioCRUD, etc.**: Formularios para crear, consultar, actualizar y eliminar registros.

Cada vista incluye:

- Componentes visuales (text fields, combo boxes, botones, tablas).
- Listeners o eventos (ActionListener) para manejar la interacción del usuario.
- Llamadas a métodos para ejecutar acciones sobre la base de datos.

b. Conector

Para establecer la conexión con la base de datos MySQL (hospedada en Clever Cloud), se descargó el **conector JDBC de MySQL** (mysql-connector-java-x.x.xx.jar) y se cargó manualmente en la carpeta de **librerías** del proyecto Java. Esta dependencia permite que el sistema interactúe con la base de datos utilizando el protocolo JDBC.

c. Conexión

El sistema se conecta a una base de datos **MySQL alojada en Clever Cloud**. Para lograrlo:

- Se descargó y configuró el **driver JDBC de MySQL**, el cual se añadió al proyecto desde las librerías.
- Se implementó la clase Conexion.java, que contiene los parámetros necesarios para establecer la conexión con el servidor en la nube (host, usuario, contraseña y puerto).

- Se utilizó un bloque try-catch para capturar posibles excepciones durante la conexión y reportar errores en consola si ocurren fallos.

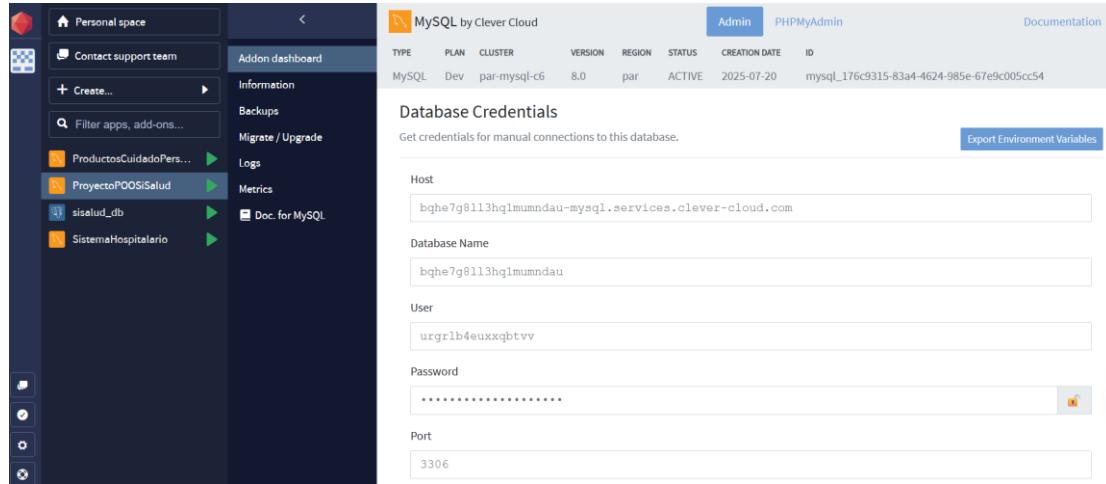


Ilustración 24. Credenciales proporcionadas por Clever Cloud para la conexión con MySQL.

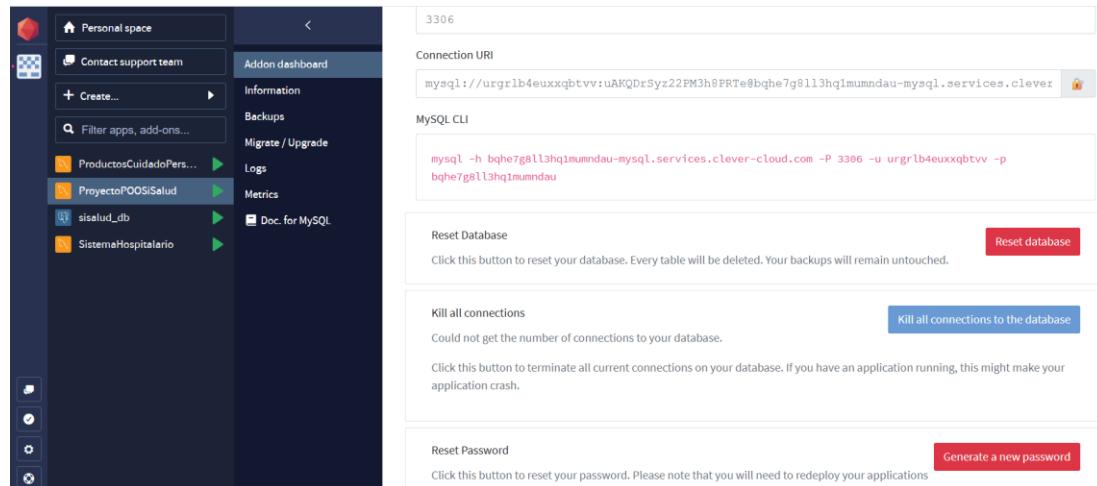


Ilustración 25. Credenciales (URI) proporcionadas por Clever Cloud para la conexión con MySQL

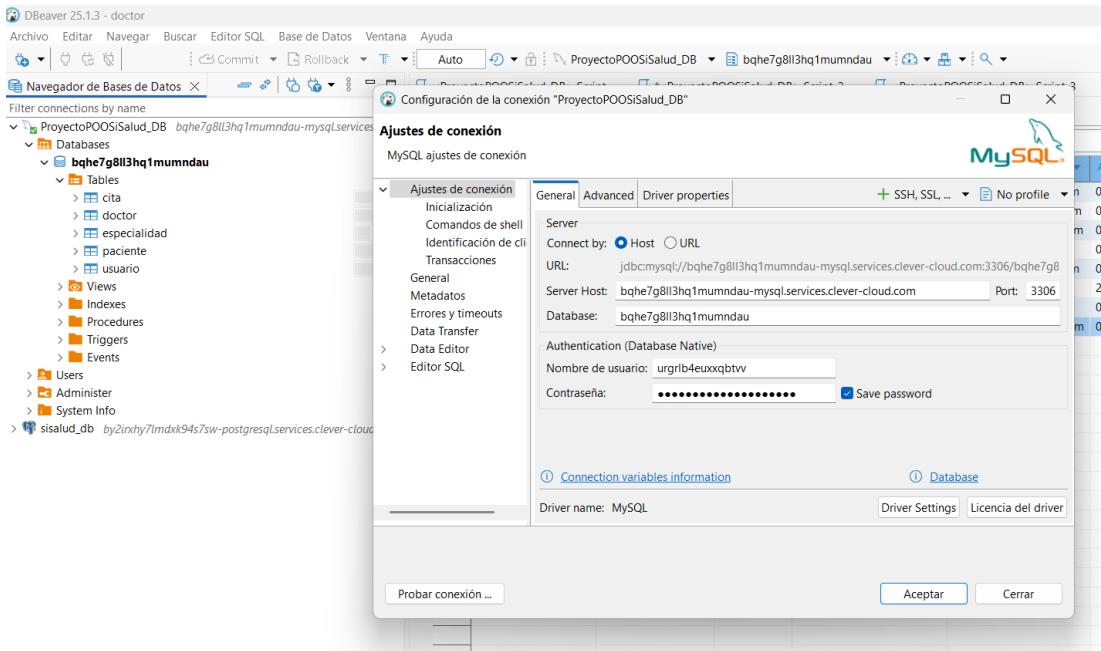


Ilustración 26. Conexión en DBeaver

d. Main

El archivo Main.java representa el punto de entrada del sistema. Desde allí se lanza el LoginForm, que permite a los usuarios ingresar al sistema de acuerdo con sus credenciales. Según el tipo de usuario (administrador o recepcionista), se carga el formulario correspondiente.

e. Ejecutable

El proceso para generar el ejecutable del sistema consta de dos etapas:

1. Generación del .jar:

- Desde el entorno de desarrollo (IntelliJ IDEA), se configura la opción "**Artifacts**" para empaquetar el proyecto completo y generar un archivo .jar.

2. Conversión a .exe:

- Se utilizó la herramienta **Launch4j 3.5** para convertir el archivo .jar en un ejecutable .exe.
- En esta herramienta se especifican:
 - La ruta del archivo .jar generado.
 - El ícono del sistema (.ico).
 - La versión mínima de Java requerida para ejecutar el sistema.

- Esto permite una distribución más sencilla del software, especialmente en equipos sin entorno de desarrollo.

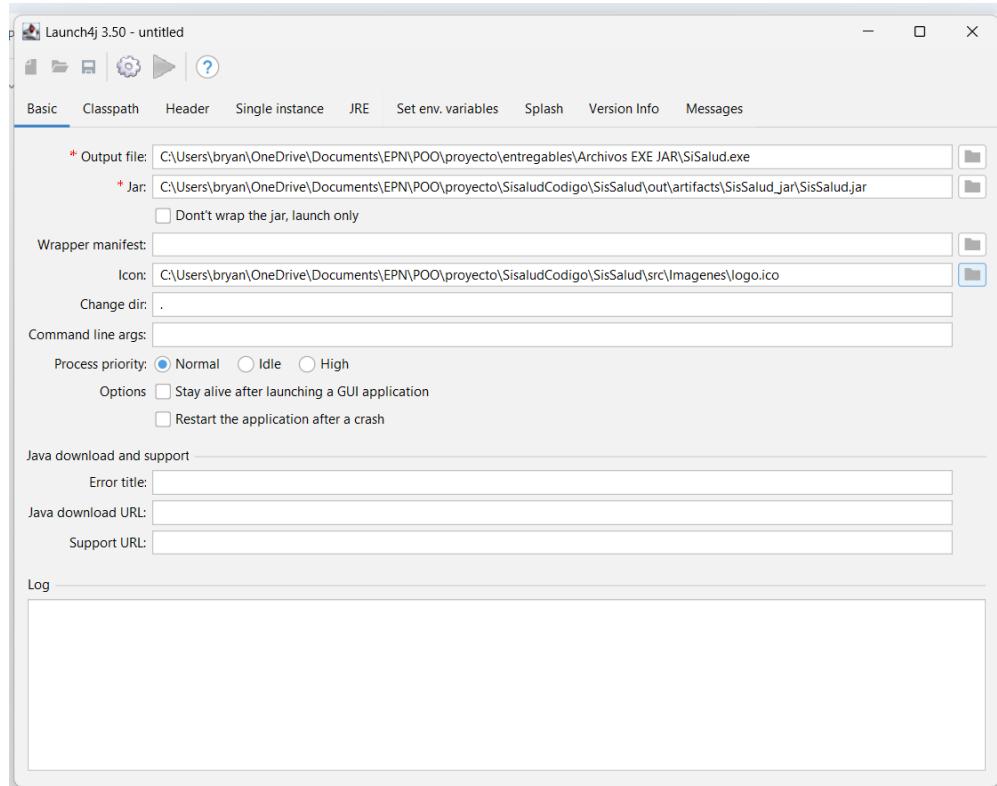


Ilustración 27. Entorno de Launch4j y configuración del ejecutable

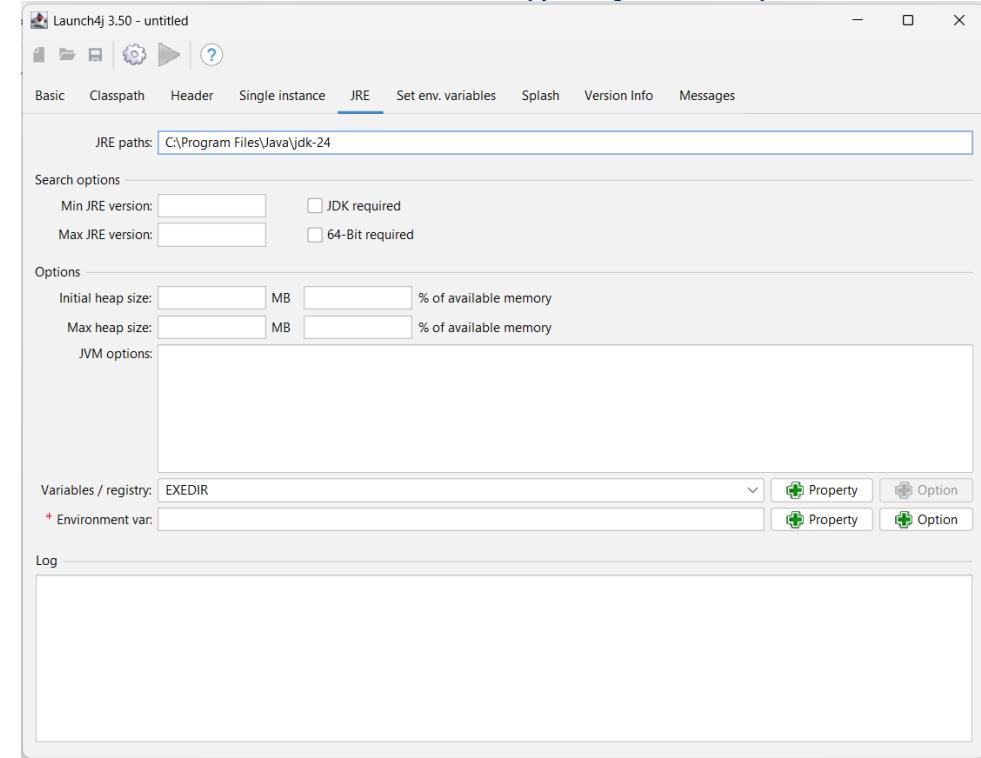


Ilustración 28. Entorno de Launch4j con la carga del JDK 24

10. Repositorio GitHub y .exe(ejecutable)

Repositorio en GIT:

<https://github.com/BryanAngulo1022/proyectoSiSalud.git>

Archivo .exe:

[https://github.com/BryanAngulo1022/proyectoSiSalud/tree/f8bd0b6649a3843a73e3785881516d6538f95605/ANEXOS%20\(informe%20y%20.exe\)/Archivo%20.EXE](https://github.com/BryanAngulo1022/proyectoSiSalud/tree/f8bd0b6649a3843a73e3785881516d6538f95605/ANEXOS%20(informe%20y%20.exe)/Archivo%20.EXE)

11. Funcionamiento con Base de Datos en la Nube

El sistema **SISALUD** está conectado a una base de datos remota desplegada en la nube mediante la plataforma **Clever Cloud**, utilizando el motor de base de datos **MySQL**. Este enfoque permite acceder a los datos desde cualquier ubicación con conexión a internet, favoreciendo la disponibilidad del sistema.

La gestión y administración de la base de datos se realizó utilizando la herramienta **DBeaver**, que facilita la conexión remota, ejecución de consultas SQL, visualización de tablas, y monitoreo en tiempo real de los registros.

Configuración de conexión

Para establecer la conexión desde la aplicación Java, se utilizó la biblioteca **JDBC (Java Database Connectivity)**, configurando la URL de conexión.

Los parámetros de acceso (host, puerto, usuario, contraseña y nombre de la base de datos) fueron proporcionados por Clever Cloud tras la creación del servicio MySQL. Estos valores se integraron en la clase de conexión del sistema (`ConexionBaseDatos`) utilizando `DriverManager.getConnection(...)`.

Funcionalidad habilitada con la base de datos en la nube

Gracias a la conexión en la nube, el sistema puede realizar operaciones en tiempo real como:

- Registro, búsqueda y edición de **pacientes, doctores y usuarios**.
- Agendamiento de **citas médicas** con validación de horarios.
- Visualización del **historial médico** de los pacientes.

- Control de **horarios ocupados** por doctor para evitar duplicidades.
- Eliminación o consulta de registros desde interfaces administrativas.

12. Conclusiones

- Durante el desarrollo del sistema SISALUD, se adquirió experiencia práctica en la planificación, diseño e implementación de un sistema real, aplicando principios fundamentales de programación orientada a objetos y la gestión de proyectos. Se comprendió la importancia de dividir el trabajo en módulos (login, registro de pacientes, agendamiento de citas, historial médico, etc.) y organizar las tareas en fases claras, facilitando el avance del proyecto. Aunque se logró una implementación funcional, se identificó que existe margen de mejora en el diseño de clases para lograr mayor cohesión, escalabilidad y reutilización futura del código.
- Se fortalecieron las habilidades en el manejo de bases de datos PostgreSQL, aprendiendo a diseñar tablas relacionales como usuario, paciente, doctor, cita, especialidad, entre otras. Se logró conectar el sistema con la base de datos externa (Clever Cloud), gestionar información de manera dinámica y realizar operaciones CRUD (crear, leer, actualizar, eliminar) desde la interfaz gráfica.
- Se aprendió a tratar información crítica como datos personales y médicos con responsabilidad, implementando validaciones y asegurando la integridad de la información mediante una adecuada estructura lógica y controles desde la interfaz.

13. Recomendaciones

- Es recomendable mantener una documentación técnica actualizada y utilizar herramientas como GitHub para versionar el código, registrar avances y colaborar en equipo.
- Se recomienda estructurar mejor el sistema usando clases específicas para cada entidad (como Paciente, Doctor, Cita, etc.) y delegar responsabilidades mediante patrones de diseño. Esto permitirá que el sistema sea más fácil de mantener, escalar y reutilizar.
- Al generar la aplicación .exe se debe verificar seleccionar versión adecuada del **JDK o JRE** correspondiente. Además, para conectar con la base de datos el archivo conector jdbc debe estar correctamente ubicado dentro del paquete en el que se va a generar el .jar o el .exe para evitar errores la ejecutarlos.

14. Anexos

Ilustración 29. Script del proyecto para la creación de la base de datos.

```
*<ProyectoPOOSiSalud_DB> Script X <ProyectoPOOSiSalud_DB> Script-2 <ProyectoPOOSiSalud_DB> Script-3 cita bqhei

-- -- Tabla USUARIO
CREATE TABLE usuario (
    id_usuario SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    usuario VARCHAR(50) unique NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL,
    contrasena VARCHAR(100) NOT NULL,
    rol VARCHAR(20) CHECK (rol IN ('Administrador', 'Repcionista'))
);

-- -- Tabla PACIENTE
CREATE TABLE paciente (
    id_paciente SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    ci VARCHAR(10) UNIQUE NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    telefono VARCHAR(15) NOT NULL,
    direccion TEXT,
    genero VARCHAR(10) CHECK (genero IN ('Masculino', 'Femenino', 'Otro')) NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL
);

-- -- Tabla ESPECIALIDAD
CREATE TABLE especialidad (
    id_especialidad SERIAL PRIMARY KEY,
    nombre VARCHAR(50)
);

-- -- Tabla DOCTOR (requiere ESPECIALIDAD)
CREATE TABLE doctor (
    id_doctor SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    ci VARCHAR(10) UNIQUE NOT null,
    id_especialidad INTEGER REFERENCES especialidad(id_especialidad),
    correo VARCHAR(100),
    telefono VARCHAR(15),
    jornada VARCHAR(20) CHECK (jornada IN ('Mañana', 'Tarde', 'Completa')) DEFAULT 'Completa',
    ci VARCHAR(10) UNIQUE NOT NULL,
    hora TIME NOT NULL,
    fecha DATE NOT NULL
);

-- -- Tabla CITA (requiere PACIENTE y DOCTOR)
CREATE TABLE cita (
    id_cita SERIAL PRIMARY KEY,
    id_paciente INTEGER NOT NULL,
    id_doctor INTEGER NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    CONSTRAINT fk_paciente FOREIGN KEY (id_paciente) REFERENCES paciente(id_paciente) ON DELETE CASCADE,
    CONSTRAINT fk_doctor FOREIGN KEY (id_doctor) REFERENCES doctor(id_doctor) ON DELETE CASCADE,
    CONSTRAINT cita_unica UNIQUE (id_doctor, fecha, hora) -- Evita duplicidad
);
```

Ilustración 30. Script para cargar datos iniciales

The screenshot shows a MySQL Workbench interface with several tabs open. The current tab displays a SQL script for populating four tables: 'usuario', 'especialidad', 'doctor', and 'cita'. The script uses the 'INSERT INTO' statement with 'VALUES' clauses to insert data into each table.

```
④ Ⓛ INSERT INTO usuario (nombre, usuario, correo, contrasena, rol)
④ Ⓜ VALUES ('Karol Fernandez', 'admin1', 'admin@sisalud.com', 'admin123', 'Administrador');
④ Ⓛ INSERT INTO usuario (nombre, usuario, correo, contrasena, rol)
④ Ⓜ VALUES ('Luis Merino', 'recep1', 'recep@sisalud.com', 'recep123', 'Repcionista');

AI Ⓛ INSERT INTO especialidad (nombre) VALUES
('Medicina General'),
('Pediatría'),
('Cardiología'),
('Ginecología'),
('Neurología');

④ Ⓛ INSERT INTO doctor (nombre, ci, id_especialidad, correo, telefono, jornada) VALUES
('Dra. Ana Torres', '1234312349', 1, 'ana.torres@hospital.com', '0987654321', 'Mañana'),
('Dr. Juan Pérez', '4939857477', 2, 'juan.perez@hospital.com', '0987456123', 'Tarde'),
('Dr. Carlos Díaz', '5431232349', 3, 'carlos.diaz@hospital.com', '0991234567', 'Completa'),
('Dra. Camila Vizcaino', '1093123249', 4, 'cam.viz@hospital.com', '0983466123', 'Tarde'),
('Dra. Ana Torres', '0312323490', 5, 'ana.torres@hospital.com', '0967845679', 'Completa');
```