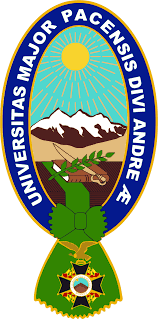
******

***UNIVERSIDAD MAYOR DE SAN ANDRES***

*FACULTA DE CIENCIAS PURAS Y NATURALES*

CARRERA DE INFORMATICA

**SISTEMA DE GESTIÓN**

**DE UN HOSPITAL**

**Integrantes:**

1.Yeral Gutiérrez Quispe

2.Luis Quelca Quelca

3.Jonathan Gutiérrez Condori

4. Grover Flores Quispe

5.Daniel Sánchez Cuentas

6.Adriana Váquela Valda

7.Raquel Serrano Mamani

**Docente:**

Lic. Rosalía Lopez

**Materia :**

INF-121 Programación II

28/01/2025

Tabla de contenido

[Tabla de Ilustraciones 3](#_Toc188958715)

[Introducción 4](#_Toc188958716)

[1. Definición del Proyecto 4](#_Toc188958717)

[1.1 Descripción General 4](#_Toc188958718)

[1.2 Objetivos 4](#_Toc188958719)

[2. Análisis y Diseño 5](#_Toc188958720)

[2.1 Diagrama UML 5](#_Toc188958721)

[2.2 Principios de Diseño 8](#_Toc188958722)

[3. Implementación en Java 12](#_Toc188958723)

[3.1 Estructura del Proyecto 12](#_Toc188958724)

[3.2 Código Fuente 14](#_Toc188958725)

[PARTE LOGICA: 14](#_Toc188958726)

[PARTE PERSISTENCIA (DAO): 30](#_Toc188958727)

[3.3 Diseño de Interfaces 38](#_Toc188958728)

[3.5 Manejo de Archivos 42](#_Toc188958729)

[3.6 Uso de Patrones de Diseño 44](#_Toc188958730)

[4. Pruebas del Sistema 45](#_Toc188958731)

[5. Conclusión 47](#_Toc188958732)

# Tabla de Ilustraciones

[Ilustración 1 Diagrama UML 5](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958733)

[Ilustración 2 Doctor UML 6](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958734)

[Ilustración 3 Enfermero UML 6](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958735)

[Ilustración 4 Hospital UML 6](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958736)

[Ilustración 5 Medicamento UML 6](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958737)

[Ilustración 6 Paciente UML 7](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958738)

[Ilustración 7 Paciente UML 7](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958739)

[Ilustración 8 Personal UML 7](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958740)

[Ilustración 9 Habitación UML 7](#_Toc188958741)

[Ilustración 10 Sala de Pacientes Estables UML 8](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958742)

[Ilustración 12 Farmacia UML 8](#_Toc188958743)

[Ilustración 11 Sala de Urgencias 8](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958744)

[Ilustración 13 Sala de Pacientes Graves UML 8](#_Toc188958745)

[Ilustración 14 Persona UML 8](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958746)

[Ilustración 15 Sala de Observación UML 8](file:///C:\Users\RAQUEL%20SERRANO\OneDrive\Documentos\InformeSistemadeGestionHospital.docx#_Toc188958747)

[Ilustración 16 Herencia de Persona UML 9](#_Toc188958748)

[Ilustración 17 Herencia de Habitación UML 10](#_Toc188958749)

[Ilustración 18 Composición de la clase Hospital UML 10](#_Toc188958750)

[Ilustración 19 Agregación de Pacientes UML 11](#_Toc188958751)

[Ilustración 20 Genericidad de medicamentos UML 11](#_Toc188958752)

[Ilustración 21 Abstraccion de Peronal UML 12](#_Toc188958753)

[Ilustración 22 Estructura En Java 13](#_Toc188958754)

[Ilustración 23 Estructura En java IGU 13](#_Toc188958755)

[Ilustración 24 Estructura en Java LOGICA 14](#_Toc188958756)

[Ilustración 25 Estructura en Java PERSISTENCIA 14](#_Toc188958757)

# Introducción

En el presente informe se describirán los pasos y aspectos importantes en el desarrollo de un proyecto que se basara en utilizar y mostrar los principios estudiados de la Programación Orientada a Objetos (POO). A demás se explicará la idea principal que se tomó para poner en práctica dichos lineamientos, que contribuyo en el desarrollo de un “Sistema de gestión de un Hospital”.

El Sistema busca optimizar la administración de un hospital mediante módulos que permiten gestionar a los pacientes, médicos, medicamentos y citas medicas de una forma eficaz , la implementación de los conceptos como la herencia, encapsulamiento y el polimorfismo fue clave para crear una buena estructura que sea funcional y fácil de mantener, garantizando un diseño escalable.

# 1. Definición del Proyecto

## 1.1 Descripción General

Adentrándonos en el desarrollo del proyecto se elaboró un “Sistema de Gestión para un Hospital”, el cual nos brindara las siguientes funcionalidades:

Ayudará en :

* El registro del hospital
* El registro de los pacientes en él sistema.
* El registro del personal en la base de datos.
* El registro de Habitaciones del Hospital
* El registro de medicamentos en el sistema de la farmacia.

Gestionará:

* Un listado de Hospitales
* La atención médica.
* La asignación de personal al paciente.
* La asignación de habitaciones del hospital.
* Un listado de los pacientes.
* Un listado del personal .
* Un listado de habitaciones.
* Un listado de medicamentos.
* La venta de medicamentos.

## 1.2 Objetivos

Se busca cumplir con los siguientes puntos para la creación de este proyecto :

* Aplicar los conceptos fundamentales de la Programación Orientada a Objetos (POO).
* Elaborar un Sistema modulable y escalable
* Implementar los patrones de diseño para mejorar la flexibilidad del código.
* Optimizar el trabajo del sistema de gestión para un Hospital.

# 2. Análisis y Diseño

## 2.1 Diagrama UML

Para realización del Sistema de Gestión para un Hospital se empleó los siguientes diagramas UML:

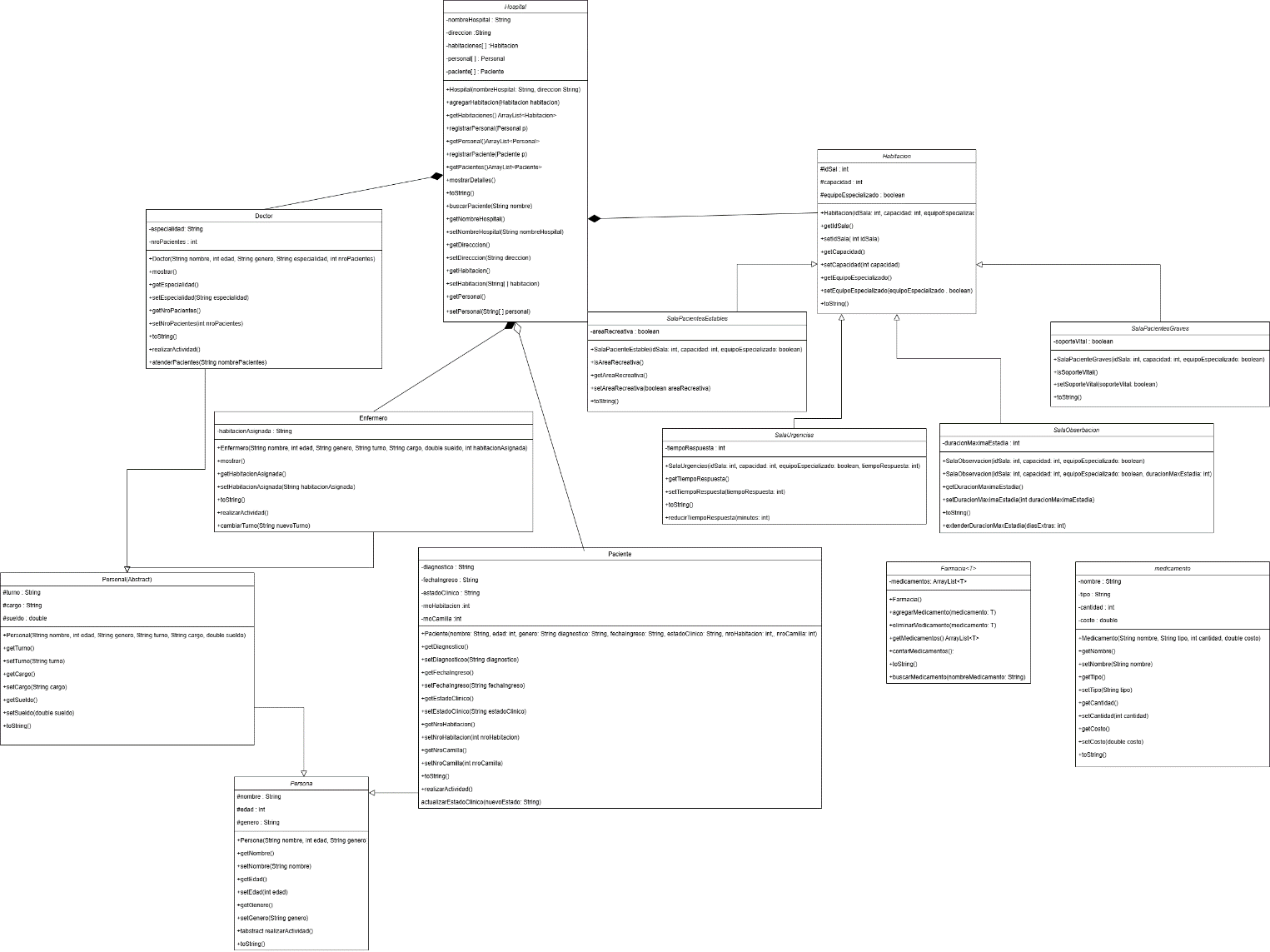


Ilustración Diagrama UML

Para un mejor análisis se tomó los siguientes diagramas de clases:

Clase Hospital:

Clase Doctor:

Ilustración Doctor UML

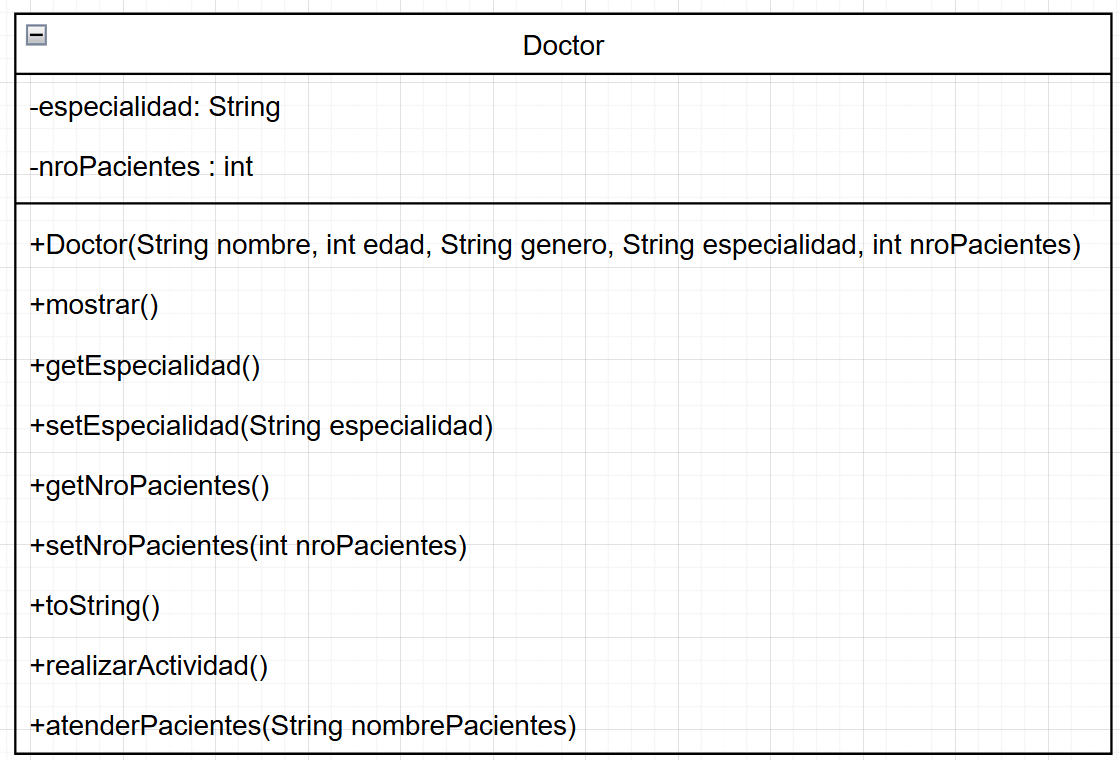
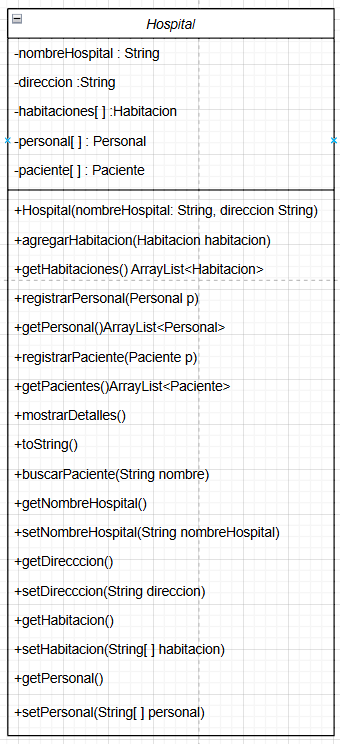


Ilustración Enfermero UML

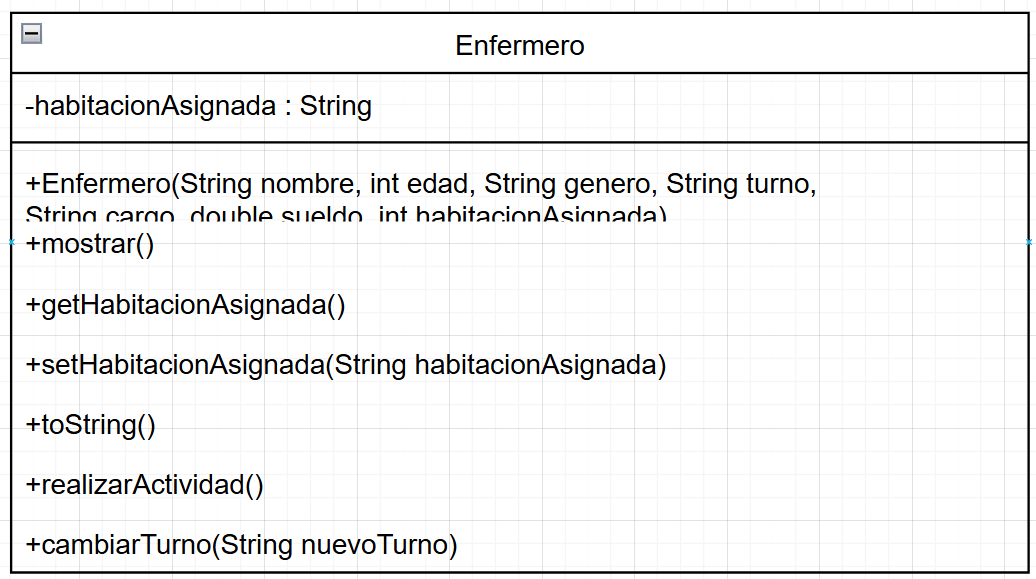
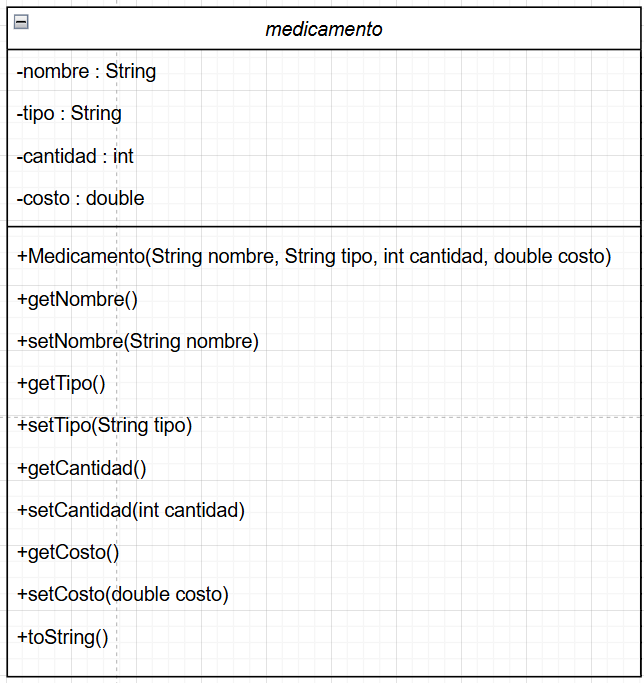
Clase Enfermero:

Ilustración Hospital UML

Clase medicamento:

Ilustración Medicamento UML



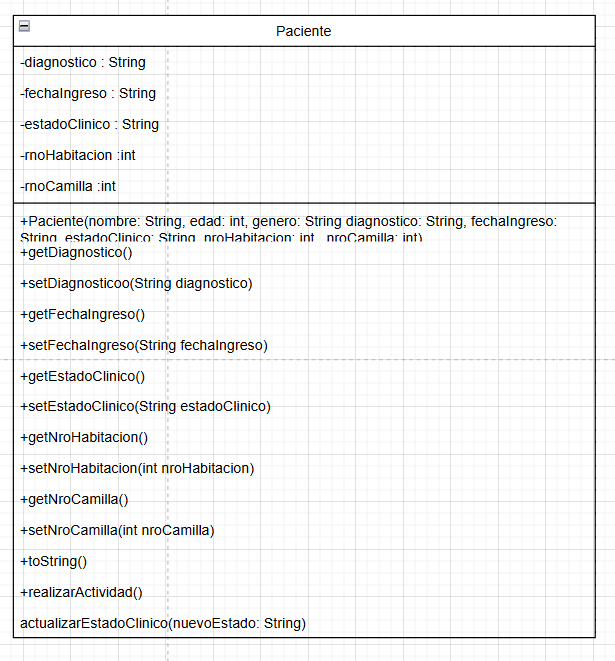
Clase Paciente:

Ilustración Paciente UML

Ilustración Paciente UML

Clase Habitación:

Clase Personal Abstracta:

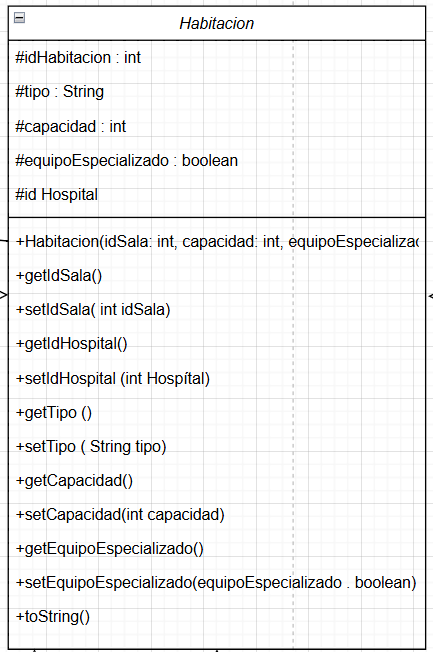
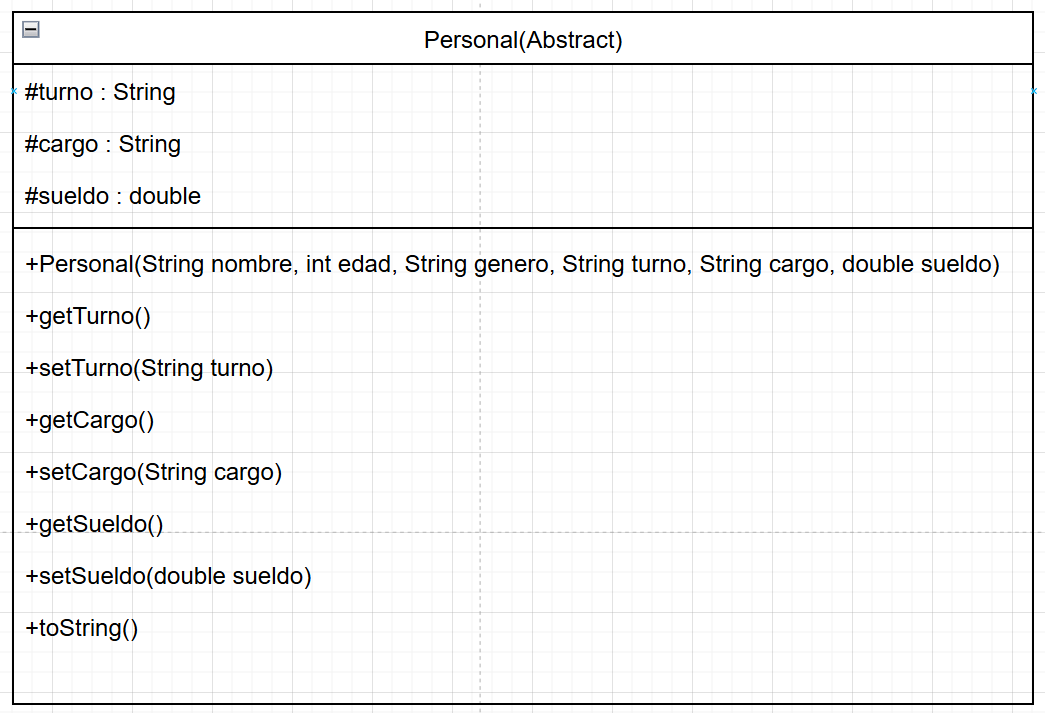


Ilustración Personal UML

Ilustración Habitación UML

Clase Farmacia Genérica:

Clase Sala de Pacientes Estables :

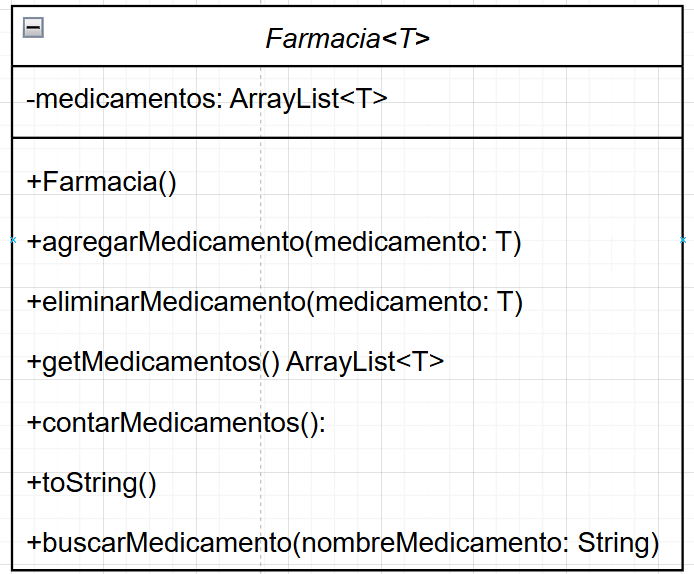
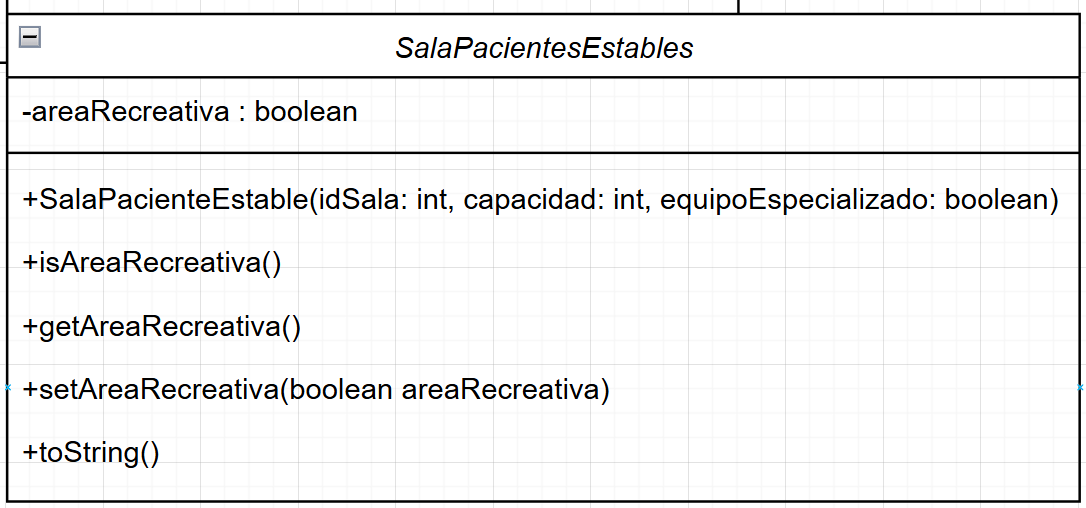


Ilustración Sala de Pacientes Estables UML

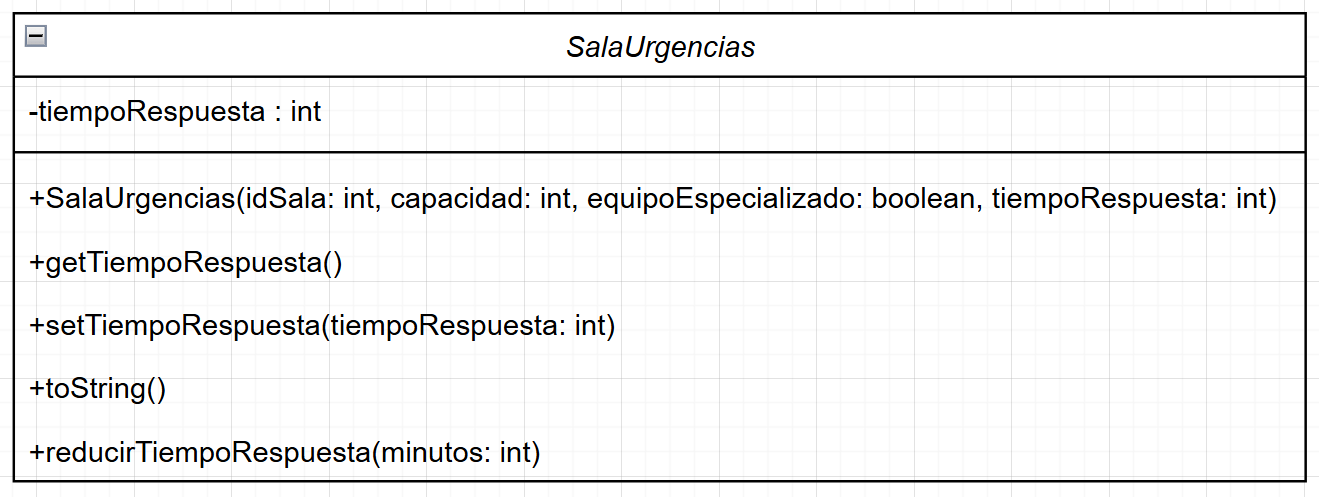
Ilustración Farmacia UML

Ilustración Sala de Urgencias

UML

Clase Sala de Pacientes Graves:

Clase Sala de Urgencias:

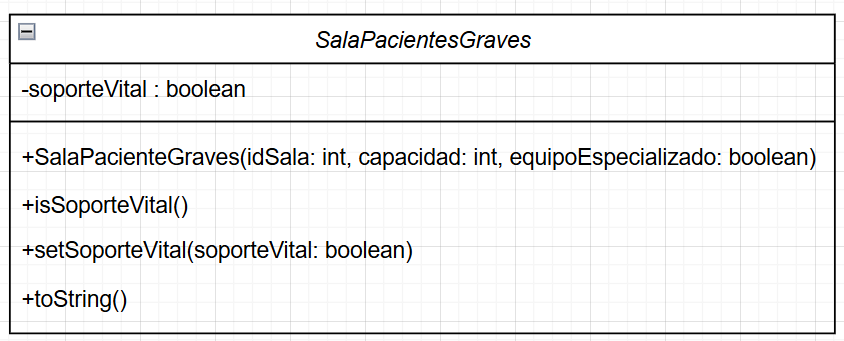


Ilustración Sala de Pacientes Graves UML

Clase Sala de Observación:

Clase Persona:

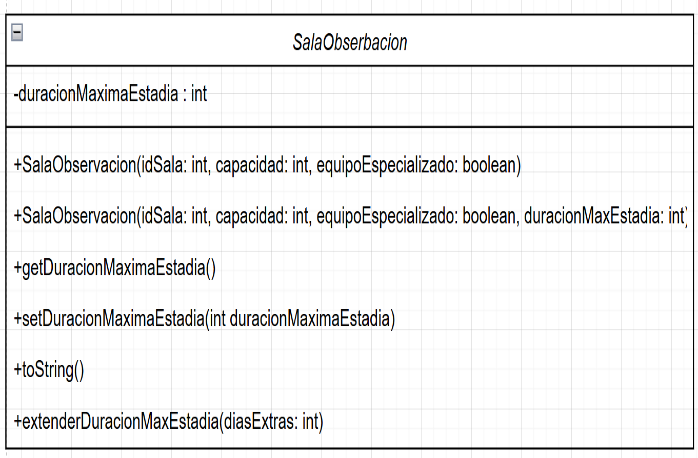
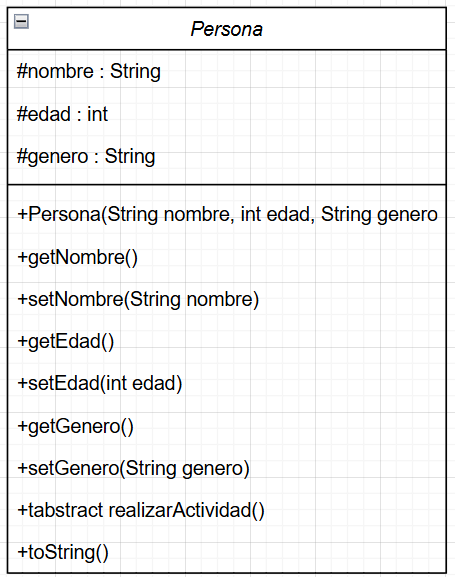


Ilustración Persona UML

Ilustración Sala de Observación UML

## 2.2 Principios de Diseño

Utiliza los siguientes conceptos de la Programación Orientada a Objetos:

Herencia:

1. Las clases , Personal, Paciente, Doctor y Enfermero heredan de Persona.

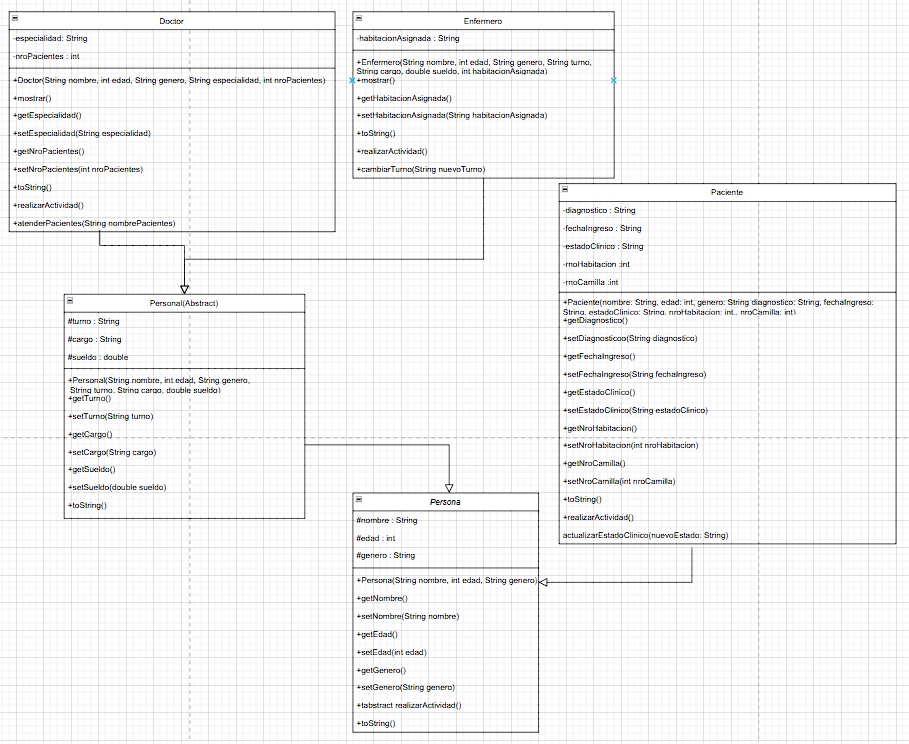
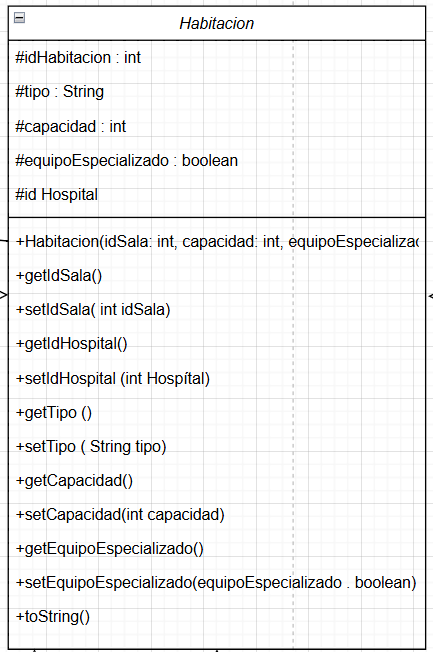


Ilustración 16 Herencia de Persona UML

1. Las clases , Sala de Pacientes Estables, Sala de Urgencias. Sala de Observación, Sala de Pacientes Graves heredan de Habitación.



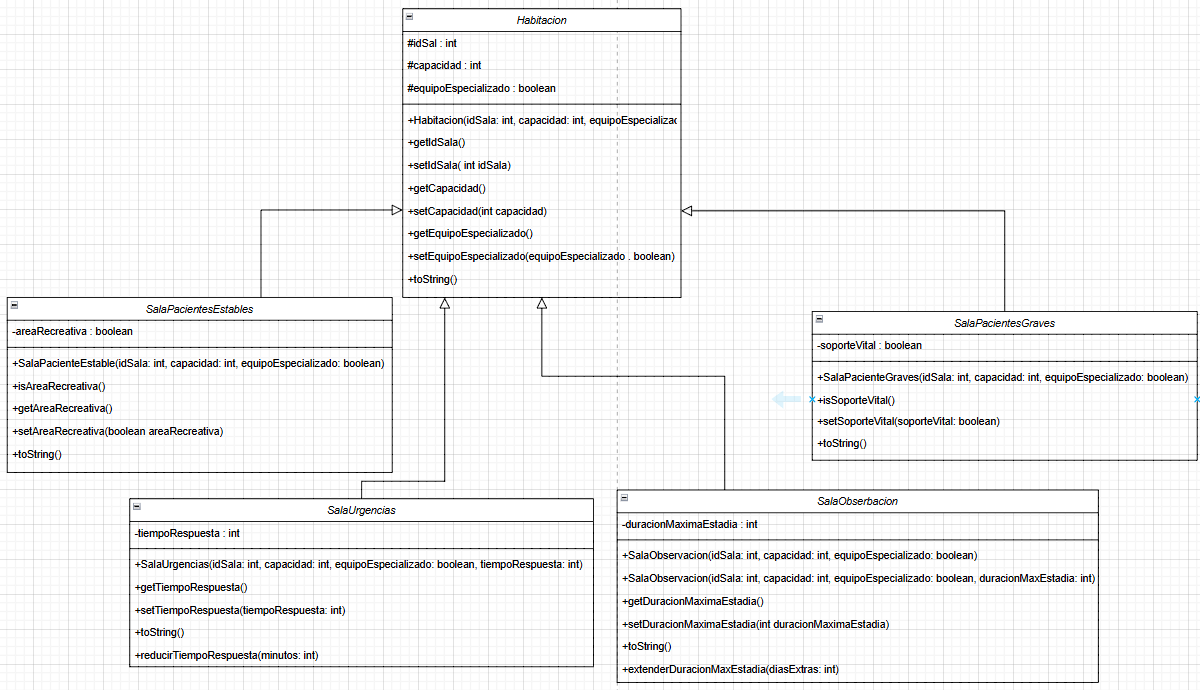


Ilustración 17 Herencia de Habitación UML

Composición: La clase Hospital incluye un objeto de tipo Habitación, un objeto de tipo Personal y un objeto de tipo Paciente.

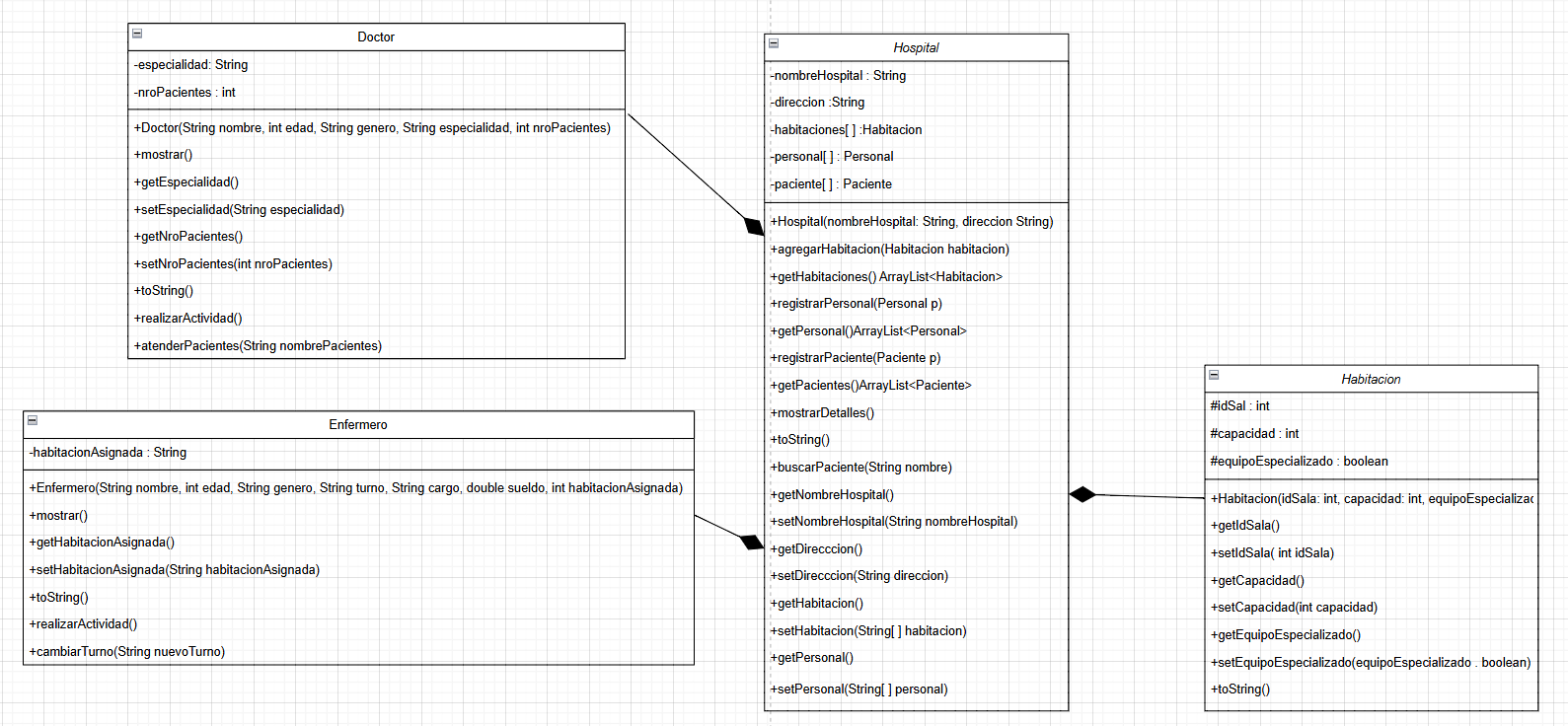


Ilustración Composición de la clase Hospital UML

Agregación: El Hospital puede estar asociado a varios Pacientes.

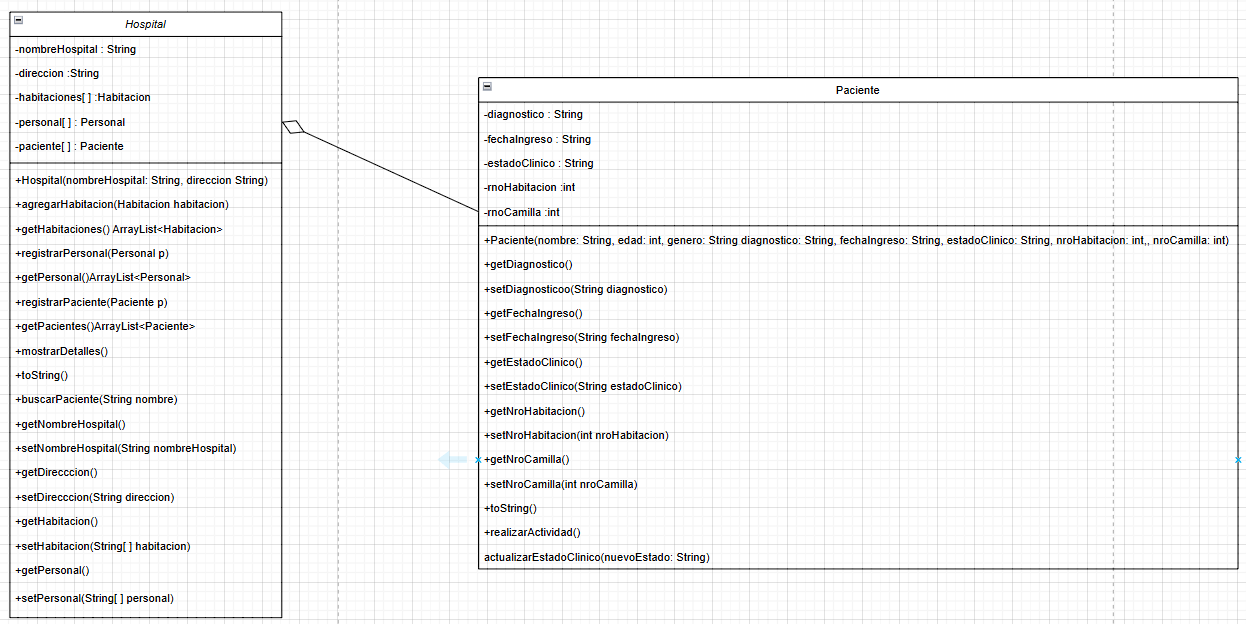


Ilustración Agregación de Pacientes UML

Genericidad: Se empleó colecciones genéricas para gestionar listas de medicamentos en Farmacia.

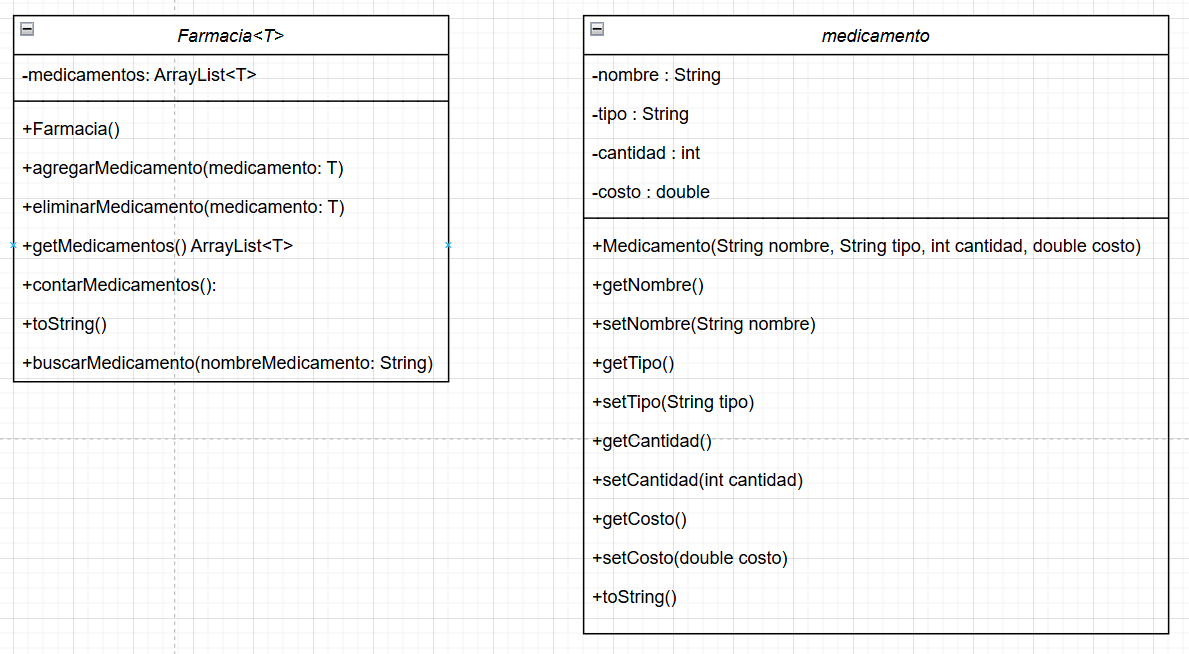


Ilustración Genericidad de medicamentos UML

Abstracción: las clases Doctor y Enfermero usaron la clase Abstracta Personal:

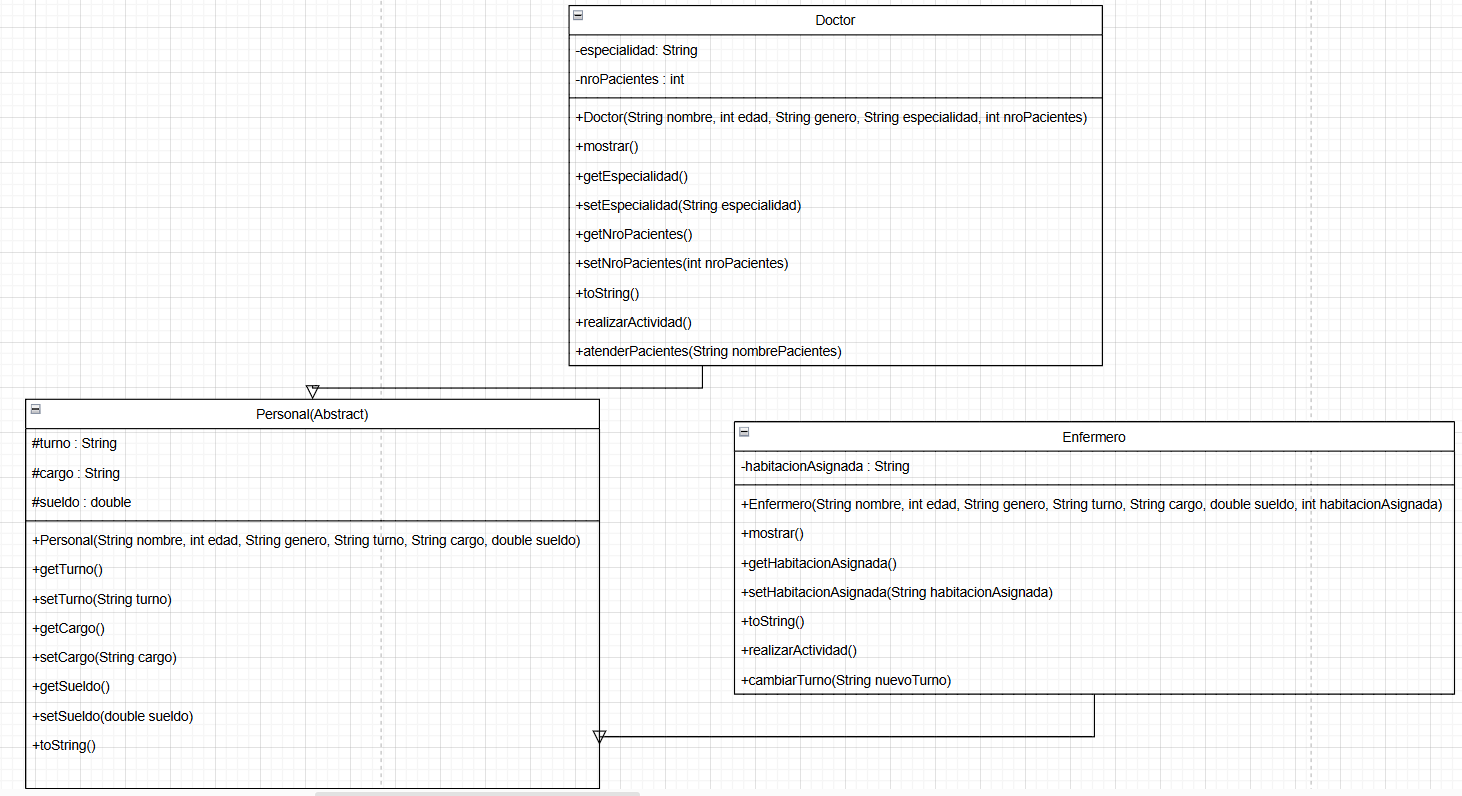


Ilustración Abstraccion de Peronal UML

# 3. Implementación en Java

## 3.1 Estructura del Proyecto

Para este proyecto se usaron los siguientes paquetes:

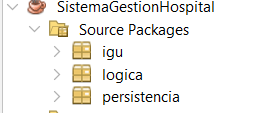


Ilustración Estructura En Java

IGU:

Este paquete se centra en la parte del diseño de la interfaz del programa, en el cual se usó los siguientes paneles:

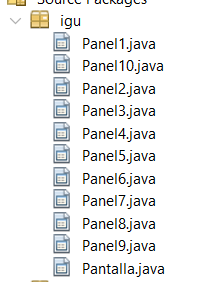


Ilustración Estructura En java IGU

LOGICA:

Se utilizo el paquete de lógica que contiene todas las clases, con sus métodos, además del principal para poder correr el programa.

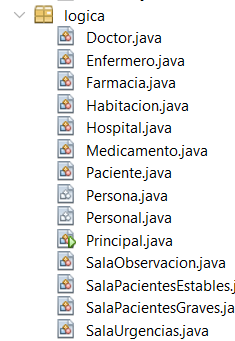


Ilustración Estructura en Java LOGICA

PERSISTENCIA:

Se utilizo el paquete “Persistencia” que contiene todo el código que funciona con la base de datos.

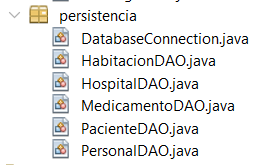


Ilustración Estructura en Java PERSISTENCIA

## 3.2 Código Fuente

Continuando con la elaboración del Sistema de Gestión de un Hospital se realizó el código en java de:

### PARTE LOGICA:

Tabla Hospital codigo Java

Clase Base: Hospital.

package logica;

import java.util.ArrayList;

public class Hospital {

private String nombreHospital;

private String direccion;

private ArrayList<Habitacion> habitaciones;

private ArrayList<Personal> personal;

private ArrayList<Paciente> pacientes;

public Hospital(String nombreHospital, String direccion) {

this.nombreHospital = nombreHospital;

this.direccion = direccion;

this.habitaciones = new ArrayList<>();

this.personal = new ArrayList<>();

this.pacientes = new ArrayList<>();

}

// gestionar habitaciones

public void agregarHabitacion(Habitacion habitacion) {

habitaciones.add(habitacion);

}

public ArrayList<Habitacion> getHabitaciones() {

return habitaciones;

}

}

// gestionar personal

public void registrarPersonal(Personal p) {

personal.add(p);

}

public ArrayList<Personal> getPersonal() {

return personal;

}

// gestionar pacientes

public void registrarPaciente(Paciente p) {

pacientes.add(p);

}

public ArrayList<Paciente> getPacientes() {

return pacientes;

}

//mostrar Detalles

public void mostrarDetalles() {

System.out.println("=== Información del Hospital ===");

System.out.println("Nombre: " + nombreHospital);

System.out.println("Dirección: " + direccion);

System.out.println("--- Habitaciones ---");

for (Habitacion h : habitaciones) {

System.out.println(h.toString());

}

System.out.println("--- Personal ---");

for (Personal p : personal) {

System.out.println(p.toString());

}

System.out.println("--- Pacientes ---");

for (Paciente p : pacientes) {

System.out.println(p.toString());

}

}

@Override

public String toString() {

return "Hospital: " + nombreHospital + "\nDirección: " + direccion + "\nHabitaciones: " + habitaciones.size() + "\nPersonal: " + personal.size() + "\nPacientes: " + pacientes.size();

}

//buscarPacientes

public Paciente buscarPaciente(String nombre) {

for (Paciente paciente : pacientes) {

if (paciente.getNombre().equalsIgnoreCase(nombre)) {

System.out.println("Paciente encontrado: " + paciente.toString());

return paciente;

}

}

System.out.println("Paciente no encontrado: " + nombre);

return null;

}

}

Clase Derivada: Paciente

package logica;

public class Paciente extends Persona {

private String diagnostico;

private String fechaIngreso;

private String estadoClinico;

private int nroHabitacion;

private int nroCamilla;

public Paciente(String nombre, int edad, String genero, String diagnostico, String fechaIngreso, String estadoClinico, int nroHabitacion, int nroCamilla) {

super(nombre, edad, genero);

this.diagnostico = diagnostico;

this.fechaIngreso = fechaIngreso;

this.estadoClinico = estadoClinico;

this.nroHabitacion = nroHabitacion;

this.nroCamilla = nroCamilla;

}

public String getDiagnostico() {

return diagnostico;

}

public void setDiagnostico(String diagnostico) {

this.diagnostico = diagnostico;

}

public String getFechaIngreso() {

return fechaIngreso;

}

public void setFechaIngreso(String fechaIngreso) {

this.fechaIngreso = fechaIngreso;

}

public String getEstadoClinico() {

return estadoClinico;

}

public void setEstadoClinico(String estadoClinico) {

this.estadoClinico = estadoClinico;

}

public int getNroHabitacion() {

return nroHabitacion;

}

public void setNroHabitacion(int nroHabitacion) {

this.nroHabitacion = nroHabitacion;

}

public int getNroCamilla() {

return nroCamilla;

}

public void setNroCamilla(int nroCamilla) {

this.nroCamilla = nroCamilla;

}

@Override

public String toString() {

return super.toString() + ", Diagnóstico: " + diagnostico + ", Fecha de Ingreso: " + fechaIngreso + ", Estado Clínico: " + estadoClinico + ", Habitación: " + nroHabitacion + ", Camilla: " + nroCamilla;

}

//actualizar Estado

public void actualizarEstadoClinico(String nuevoEstado) {

System.out.println("El estado clínico del paciente " + this.getNombre() + " ha cambiado a: " + nuevoEstado);

this.estadoClinico = nuevoEstado;

}

public void realizarActividad() {

System.out.println("El paciente indica susu sintomas.");

}

}

Clase Derivada: Doctor.

package logica;

public class Doctor extends Personal {

private String especialidad;

private int nroPacientes;

public Doctor(String nombre, int edad, String genero, String especialidad, int nroPacientes) {

super(nombre, edad, genero, "Diurno", "Doctor", 5000.00);

this.especialidad = especialidad;

this.nroPacientes = nroPacientes;

}

public String getEspecialidad() {

return especialidad;

}

public void setEspecialidad(String especialidad) {

this.especialidad = especialidad;

}

public int getNroPacientes() {

return nroPacientes;

}

public void setNroPacientes(int nroPacientes) {

this.nroPacientes = nroPacientes;

}

public void atenderPaciente(String nombrePaciente) {

System.out.println("El doctor " + this.getNombre() + " está atendiendo al paciente: " + nombrePaciente);

this.nroPacientes++;

}

@Override

public String toString() {

return super.toString() + ", Especialidad: " + especialidad + ", Número de Pacientes: " + nroPacientes;

}

public void realizarActividad() {

System.out.println("El doctor realiza un diagnóstico.");

}

}

Clase Derivada: Enfermero.

package logica;

public class Enfermero extends Personal {

private int habitacionAsignada;

public Enfermero(String nombre, int edad, String genero, String turno, String cargo, double sueldo, int habitacionAsignada) {

super(nombre, edad, genero, turno, cargo, sueldo);

this.habitacionAsignada = habitacionAsignada;

}

public int getHabitacionAsignada() {

return habitacionAsignada;

}

public void setHabitacionAsignada(int habitacionAsignada) {

this.habitacionAsignada = habitacionAsignada;

}

@Override

public String toString() {

return super.toString() + ", Habitación Asignada: " + habitacionAsignada;

}

//cambiarTurno

public void cambiarTurno(String nuevoTurno) {

System.out.println("El turno del enfermero " + this.getNombre() + " ha sido cambiado a: " + nuevoTurno);

this.setTurno(nuevoTurno);

}

public void realizarActividad() {

System.out.println("El enfermero habla con un paciente");

}

}

Clase Derivada Habitación.

package logica;

public class Habitacion {

private int idHabitacion;

private String tipo;

private int capacidad;

private boolean equipoEspecializado;

private int idHospital;

public Habitacion(int idHabitacion,String tipo, int capacidad, boolean equipoEspecializado, int idHospital) {

this.idHabitacion = idHabitacion;

this.tipo = tipo;

this.capacidad = capacidad;

this.equipoEspecializado = equipoEspecializado;

this.idHospital = idHospital;

}

public int getIdSala() {

return idHabitacion;

}

public int getIdHospital() {

return idHospital;

}

public void setIdHospital(int idHospital) {

this.idHospital = idHospital;

}

public int getIdHabitacion() {

return idHabitacion;

}

public void setIdHabitacion(int idHabitacion) {

this.idHabitacion = idHabitacion;

}

public String getTipo() {

return tipo;

}

public void setTipo(String tipo) {

this.tipo = tipo;

}

public void setIdSala(int idSala) {

this.idHabitacion = idSala;

}

public int getCapacidad() {

return capacidad;

}

public void setCapacidad(int capacidad) {

this.capacidad = capacidad;

}

public boolean isEquipoEspecializado() {

return equipoEspecializado;

}

public void setEquipoEspecializado(boolean equipoEspecializado) {

this.equipoEspecializado = equipoEspecializado;

}

@Override

public String toString() {

return "Habitacion [ID Sala: " + idHabitacion + ", Capacidad: " + capacidad + ", Equipo Especializado: " + (equipoEspecializado ? "Sí" : "No") + "]";

}

}

Clase Derivada: Medicamento.

package logica;

public class Medicamento {

private String nombre;

private String tipo;

private int cantidad;

private double costo;

public Medicamento(String nombre, String tipo, int cantidad, double costo) {

this.nombre = nombre;

this.tipo = tipo;

this.cantidad = cantidad;

this.costo = costo;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getTipo() {

return tipo;

}

public void setTipo(String tipo) {

this.tipo = tipo;

}

public int getCantidad() {

return cantidad;

}

public void setCantidad(int cantidad) {

this.cantidad = cantidad;

}

public double getCosto() {

return costo;

}

public void setCosto(double costo) {

this.costo = costo;

}

@Override

public String toString() {

return "Medicamento [Nombre: " + nombre + ", Tipo: " + tipo + ", Cantidad: " + cantidad + ", Costo: Bs." + costo + "]";

}

}

Clase Derivada: Persona.

package logica;

public abstract class Persona {

protected String nombre;

protected int edad;

protected String genero;

public Persona(String nombre, int edad, String genero) {

this.nombre = nombre;

this.edad = edad;

this.genero = genero;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public int getEdad() {

return edad;

}

public void setEdad(int edad) {

this.edad = edad;

}

public String getGenero() {

return genero;

}

public void setGenero(String genero) {

this.genero = genero;

}

@Override

public String toString() {

return "Persona [Nombre: " + nombre + ", Edad: " + edad + ", Género: " + genero + "]";

}

public abstract void realizarActividad();

}

Clase Derivada Personal.

package logica;

public abstract class Personal extends Persona {

protected String turno;

protected String cargo;

protected double sueldo;

public Personal(String nombre, int edad, String genero, String turno, String cargo, double sueldo) {

super(nombre, edad, genero);

this.turno = turno;

this.cargo = cargo;

this.sueldo = sueldo;

}

public String getTurno() {

return turno;

}

public void setTurno(String turno) {

this.turno = turno;

}

public String getCargo() {

return cargo;

}

public void setCargo(String cargo) {

this.cargo = cargo;

}

public double getSueldo() {

return sueldo;

}

public void setSueldo(double sueldo) {

this.sueldo = sueldo;

}

@Override

public String toString() {

return super.toString() + ", Turno: " + turno + ", Cargo: " + cargo + ", Sueldo: Bs." + sueldo;

}

}

Clase Derivada Farmacia (Abstracta)

package logica;

import java.util.ArrayList;

public class Farmacia<T> {

private ArrayList<T> medicamentos;

public Farmacia() {

this.medicamentos = new ArrayList<>();

}

public void agregarMedicamento(T medicamento) {

medicamentos.add(medicamento);

}

public void eliminarMedicamento(T medicamento) {

medicamentos.remove(medicamento);

}

public ArrayList<T> getMedicamentos() {

return medicamentos;

}

public int contarMedicamentos() {

return medicamentos.size();

}

@Override

public String toString() {

return "Farmacia [Medicamentos almacenados: " + medicamentos.size() + "]";

}

//buscarMedicamento

public T buscarMedicamento(String nombreMedicamento) {

for (T medicamento : medicamentos) {

if (medicamento.toString().contains(nombreMedicamento)) {

System.out.println("Medicamento encontrado: " + medicamento.toString());

return medicamento;

}

}

System.out.println("Medicamento no encontrado: " + nombreMedicamento);

return null;

}

}

Clase Derivada Sala de Observación.

package logica;

public class SalaObservacion extends Habitacion {

private int duracionMaxEstadia;

public SalaObservacion(int idHabitacion, String tipo, int capacidad, boolean equipoEspecializado, int idHospital) {

super(idHabitacion ,tipo, capacidad ,equipoEspecializado ,idHospital);

this.duracionMaxEstadia = 0;

}

public SalaObservacion(int idHabitacion, String tipo, int capacidad, boolean equipoEspecializado, int idHospital, int duracionMaxEstadia) {

super(idHabitacion, tipo ,capacidad, equipoEspecializado, idHospital);

this.duracionMaxEstadia = duracionMaxEstadia;

}

public int getDuracionMaxEstadia() {

return duracionMaxEstadia;

}

public void setDuracionMaxEstadia(int duracionMaxEstadia) {

this.duracionMaxEstadia = duracionMaxEstadia;

}

@Override

public String toString() {

return super.toString() + ", Duración Máxima de Estadía: " + duracionMaxEstadia + " días";

}

public void extenderDuracionMaxEstadia(int diasExtras) {

if (diasExtras > 0) {

duracionMaxEstadia += diasExtras;

System.out.println("La duración máxima de estadía ha sido extendida a: " + duracionMaxEstadia + " días.");

} else {

System.out.println("Los días extras deben ser positivos.");

}

}

}

}

Clase Derivada Sala de Pacientes Estables.

package logica;

public class SalaPacientesEstables extends Habitacion {

private boolean areaRecreativa;

public SalaPacientesEstables(int idHabitacion, String tipo, int capacidad, boolean equipoEspecializado, int idHospital) {

super(idHabitacion,tipo,capacidad,equipoEspecializado,idHospital);

this.areaRecreativa = false;

}

public boolean isAreaRecreativa() {

return areaRecreativa;

}

public void setAreaRecreativa(boolean areaRecreativa) {

this.areaRecreativa = areaRecreativa;

}

@Override

public String toString() {

return super.toString() + ", Area Recreativa: " + (areaRecreativa ? "Sí" : "No");

}

}

Clase Derivada Sala de Pacientes Graves.

package logica;

public class SalaPacientesGraves extends Habitacion {

private boolean soporteVital;

public SalaPacientesGraves(int idHabitacion, String tipo, int capacidad, boolean equipoEspecializado, int idHospital) {

super(idHabitacion,tipo,capacidad, equipoEspecializado,idHospital);

this.soporteVital = true;

}

public boolean isSoporteVital() {

return soporteVital;

}

public void setSoporteVital(boolean soporteVital) {

this.soporteVital = soporteVital;

}

@Override

public String toString() {

return super.toString() + ", Soporte Vital: " + (soporteVital ? "Sí" : "No");

}

}

Clase Derivada Sala de Urgencias.

package logica;

public class SalaUrgencias extends Habitacion {

private int tiempoRespuesta;

public SalaUrgencias(int idHabitacion, String tipo, int capacidad, boolean equipoEspecializado, int idHospital, int tiempoRespuesta) {

super(idHabitacion, tipo, capacidad, equipoEspecializado, idHospital);

this.tiempoRespuesta = tiempoRespuesta;

}

public int getTiempoRespuesta() {

return tiempoRespuesta;

}

public void setTiempoRespuesta(int tiempoRespuesta) {

this.tiempoRespuesta = tiempoRespuesta;

}@Override

public String toString() {

return super.toString() + ", Tiempo de Respuesta: " + tiempoRespuesta + " minutos";

}

//reducirTiempo

public void reducirTiempoRespuesta(int minutos) {

if (minutos > 0 && tiempoRespuesta - minutos >= 0) {

tiempoRespuesta -= minutos;

System.out.println("El tiempo de respuesta de la sala de urgencias ha sido reducido a: " + tiempoRespuesta + " minutos");

} else {

System.out.println("El tiempo de respuesta no puede ser reducido más allá de 0.");

}

}

}

#### Clase Controlador: Principal.

package logica;

import igu.Pantalla;

import javax.swing.SwingUtilities;

import persistencia.DatabaseConnection;

import persistencia.HabitacionDAO;

import persistencia.HospitalDAO;

import persistencia.MedicamentoDAO;

import persistencia.PacienteDAO;

import persistencia.PersonalDAO;

public class Principal {

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() { //mas de lo mismo

Pantalla pantallaUsuario = new Pantalla();

pantallaUsuario.setVisible(true);

pantallaUsuario.setLocationRelativeTo(null);

}

});

HospitalDAO hospitalDAO = new HospitalDAO();

PacienteDAO pacienteDAO = new PacienteDAO();

HabitacionDAO habitacionDAO = new HabitacionDAO();

PersonalDAO personalDAO = new PersonalDAO();

MedicamentoDAO medicamentoDAO = new MedicamentoDAO();

if (DatabaseConnection.getConnection() != null) {

System.out.println("Conexión establecida correctamente.");

} else {

System.out.println("No se pudo establecer la conexión.");

}

}

}

### PARTE PERSISTENCIA (DAO):

Data Base Connection:

package persistencia;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String URL = "jdbc:mysql://sql10.freesqldatabase.com:3306/sql10759551";

private static final String USER = "sql10759551";

private static final String PASSWORD = "lLfwUXLlJP";

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 con la base de datos: " + e.getMessage());

}

return connection;

}

}

Habitacion DAO:

package persistencia;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import logica.Habitacion;

public class HabitacionDAO {

public void insertarHabitacion(String tipo, int capacidad, boolean equipoEspecializado, int idHospital) {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "INSERT INTO habitacion (id\_habitacion,tipo, capacidad, equipo\_especializado, id\_hospital) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, tipo);

statement.setInt(2, capacidad);

statement.setBoolean(3, equipoEspecializado);

statement.setInt(4, idHospital);

statement.executeUpdate();

System.out.println("Habitación insertada correctamente.");

} catch (SQLException e) {

System.out.println("Error al insertar habitación: " + e.getMessage());

}}

public List<Habitacion> listarHabitaciones() {

List<Habitacion> habitaciones = new ArrayList<>();

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "SELECT \* FROM habitacion";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

Habitacion habitacion = new Habitacion(

resultSet.getInt("id\_habitacion"),

resultSet.getString("tipo"),

resultSet.getInt("capacidad"),

resultSet.getBoolean("equipo\_especializado"),

resultSet.getInt("id\_hospital")

);

habitaciones.add(habitacion); }

} catch (SQLException e) {

System.out.println("Error al listar habitaciones: " + e.getMessage());

}

return habitaciones;

}

}

Hospital DAO:

package persistencia;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class HospitalDAO {

public void insertarHospital(String nombre, String direccion) {

// Validaciones al inicio

if (nombre == null || nombre.isEmpty()) {

System.out.println("Error: El nombre del hospital no puede estar vacío.");

return;

}

if (direccion == null || direccion.isEmpty()) {

System.out.println("Error: La dirección del hospital no puede estar vacía.");

return;

}

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "INSERT INTO hospital (nombre, direccion) VALUES (?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, nombre);

statement.setString(2, direccion);

statement.executeUpdate();

System.out.println("Hospital insertado correctamente.");

} catch (SQLException e) {

System.out.println("Error al insertar hospital: " + e.getMessage());

}

}

public void listarHospitales() {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "SELECT \* FROM hospital";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

System.out.println("ID: " + resultSet.getInt("id\_hospital"));

System.out.println("Nombre: " + resultSet.getString("nombre"));

System.out.println("Dirección: " + resultSet.getString("direccion"));

System.out.println("------------------------");

}

} catch (SQLException e) {

System.out.println("Error al listar hospitales: " + e.getMessage());

}

}

}

public void listarHospitales() {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "SELECT \* FROM hospital";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

System.out.println("ID: " + resultSet.getInt("id\_hospital"));

System.out.println("Nombre: " + resultSet.getString("nombre"));

System.out.println("Dirección: " + resultSet.getString("direccion"));

System.out.println("------------------------");

}

} catch (SQLException e) {

System.out.println("Error al listar hospitales: " + e.getMessage());

}

}

}

Medicamento DAO:

package persistencia;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class MedicamentoDAO {

public void insertarMedicamento(String nombre, String tipo, int cantidad, double costo, int idHospital) {

if (nombre == null || nombre.isEmpty()) {

System.out.println("Error: El nombre del medicamento no puede estar vacío.");

return;

}

if (tipo == null || tipo.isEmpty()) {

System.out.println("Error: El tipo de medicamento no puede estar vacío.");

return;

}

if (cantidad <= 0) {

System.out.println("Error: La cantidad debe ser un valor positivo.");

return;

}

if (costo <= 0) {

System.out.println("Error: El costo debe ser un valor positivo.");

return;

}

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "INSERT INTO medicamento (nombre, tipo, cantidad, costo, id\_hospital) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, nombre);

statement.setString(2, tipo);

statement.setInt(3, cantidad);

statement.setDouble(4, costo);

statement.setInt(5, idHospital);

statement.executeUpdate();

System.out.println("Medicamento insertado correctamente.");

} catch (SQLException e) {

System.out.println("Error al insertar medicamento: " + e.getMessage());

}}

Paciente DAO:

package persistencia;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class PacienteDAO {

public void insertarPaciente(String nombre, int edad, String genero, String diagnostico, String fechaIngreso, String estadoClinico, int idHabitacion) {

if (nombre == null || nombre.isEmpty()) {

System.out.println("Error: El nombre del paciente no puede estar vacío.");

return;

}

if (edad <= 0 || edad > 120) {

System.out.println("Error: La edad debe ser un valor positivo y razonable.");

return;

}

if (!genero.equalsIgnoreCase("Masculino") && !genero.equalsIgnoreCase("Femenino")) {

System.out.println("Error: El género debe ser 'Masculino' o 'Femenino'.");

return;

}

if (fechaIngreso == null || fechaIngreso.isEmpty()) {

System.out.println("Error: La fecha de ingreso no puede estar vacía.");

return;

}

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "INSERT INTO paciente (nombre, edad, genero, diagnostico, fecha\_ingreso, estado\_clinico, id\_habitacion) VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, nombre);

statement.setInt(2, edad);

statement.setString(3, genero);

statement.setString(4, diagnostico);

statement.setString(5, fechaIngreso);

statement.setString(6, estadoClinico);

statement.setInt(7, idHabitacion);

statement.executeUpdate();

System.out.println("Paciente insertado correctamente.");

} catch (SQLException e) {

System.out.println("Error al insertar paciente: " + e.getMessage());}}

public void listarPacientes() {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "SELECT \* FROM paciente";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

System.out.println("ID: " + resultSet.getInt("id\_paciente"));

System.out.println("Nombre: " + resultSet.getString("nombre"));

System.out.println("Edad: " + resultSet.getInt("edad"));

System.out.println("Género: " + resultSet.getString("genero"));

System.out.println("Diagnóstico: " + resultSet.getString("diagnostico"));

System.out.println("Fecha de Ingreso: " + resultSet.getString("fecha\_ingreso"));

System.out.println("Estado Clínico: " + resultSet.getString("estado\_clinico"));

System.out.println("------------------------");

}

} catch (SQLException e) {

System.out.println("Error al listar pacientes: " + e.getMessage());

}

}

}

Personal DAO:

package persistencia;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class PersonalDAO {

public void insertarPersonal(String nombre, int edad, String genero, String turno, String cargo, double sueldo, String especialidad, int idHospital) {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "INSERT INTO personal (nombre, edad, genero, turno, cargo, sueldo, especialidad, id\_hospital) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, nombre);

statement.setInt(2, edad);

statement.setString(3, genero);

statement.setString(4, turno);

statement.setString(5, cargo);

statement.setDouble(6, sueldo);

statement.setString(7, especialidad);

statement.setInt(8, idHospital);

statement.executeUpdate();

System.out.println("Personal insertado correctamente.");

} catch (SQLException e) {

System.out.println("Error al insertar personal: " + e.getMessage()); } }

public void listarPersonal() {

try (Connection connection = DatabaseConnection.getConnection()) {

String sql = "SELECT \* FROM personal";

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

System.out.println("ID: " + resultSet.getInt("id\_personal"));

System.out.println("Nombre: " + resultSet.getString("nombre"));

System.out.println("Edad: " + resultSet.getInt("edad"));

System.out.println("Género: " + resultSet.getString("genero"));

System.out.println("Turno: " + resultSet.getString("turno"));

System.out.println("Cargo: " + resultSet.getString("cargo"));

System.out.println("Sueldo: " + resultSet.getDouble("sueldo"));

System.out.println("Especialidad: " + resultSet.getString("especialidad"));

System.out.println("ID Hospital: " + resultSet.getInt("id\_hospital"));

System.out.println("------------------------"); }

} catch (SQLException e) {

System.out.println("Error al listar personal: " + e.getMessage());

} }}

## 3.3 Diseño de Interfaces

La primera parte de la interfaz del proyecto es la siguiente ventana:

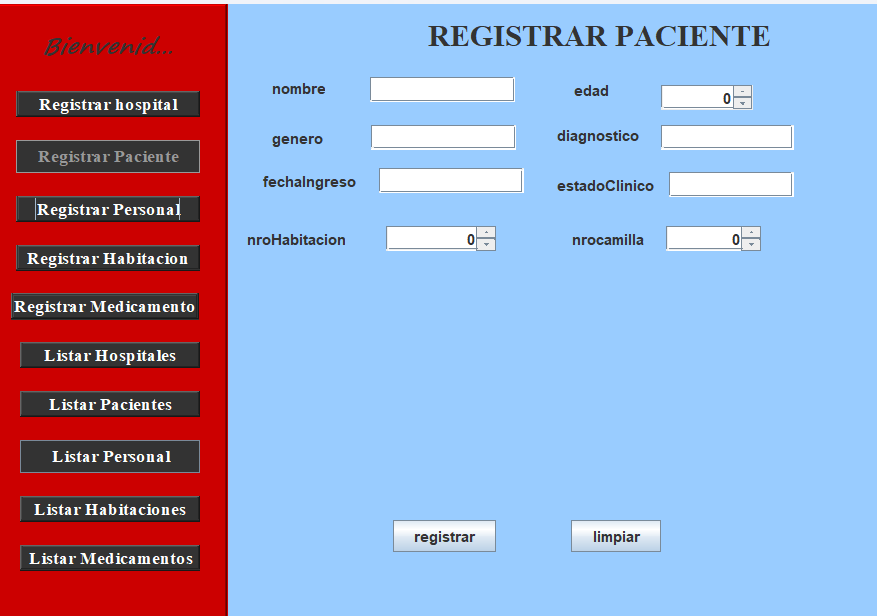
La Ventana Principal:



La siguiente ventana es Registra Hospital:



Registrar Paciente:



Registrar Personal:



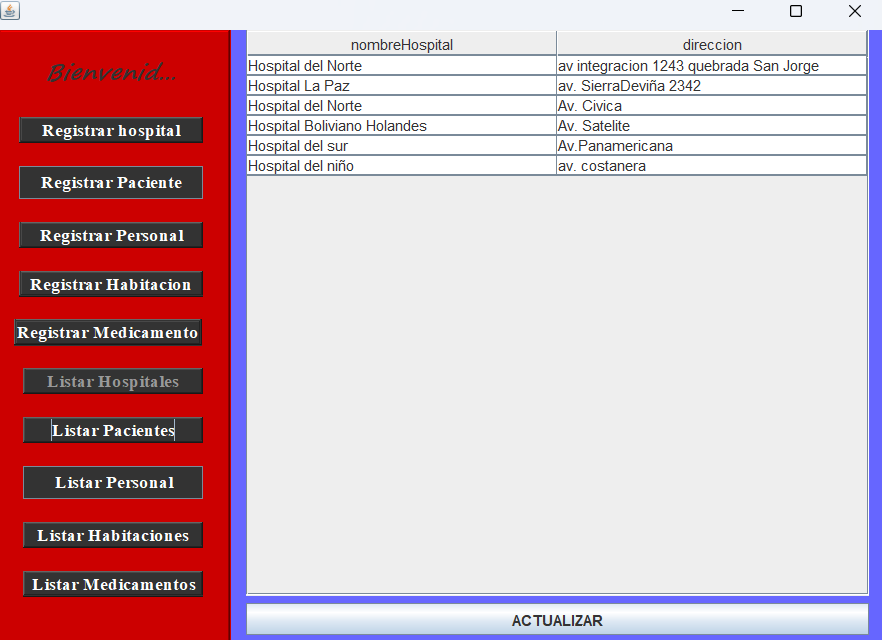
Registrar Habitación:



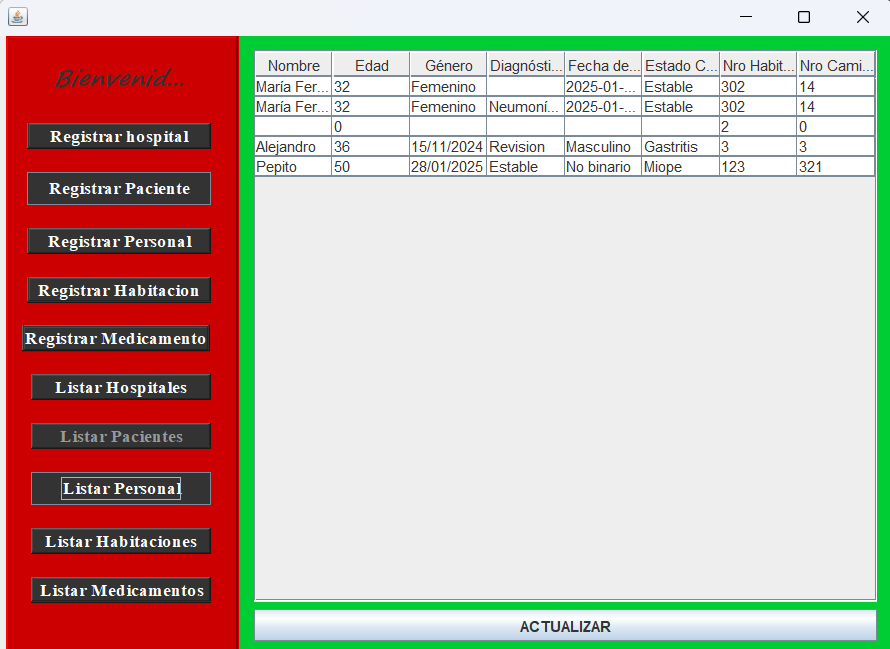
Registro Medicamento:



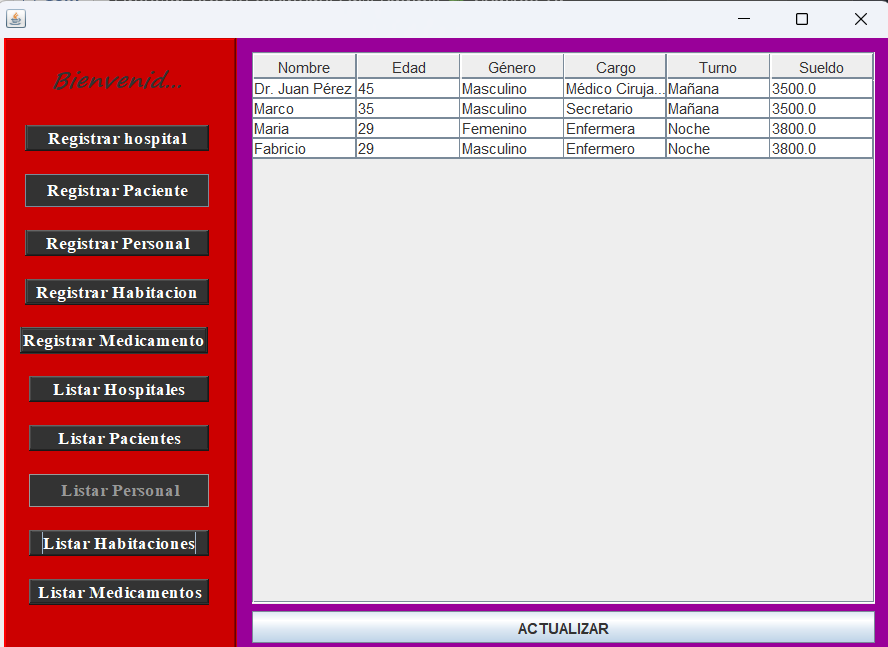
Listar Hospitales:



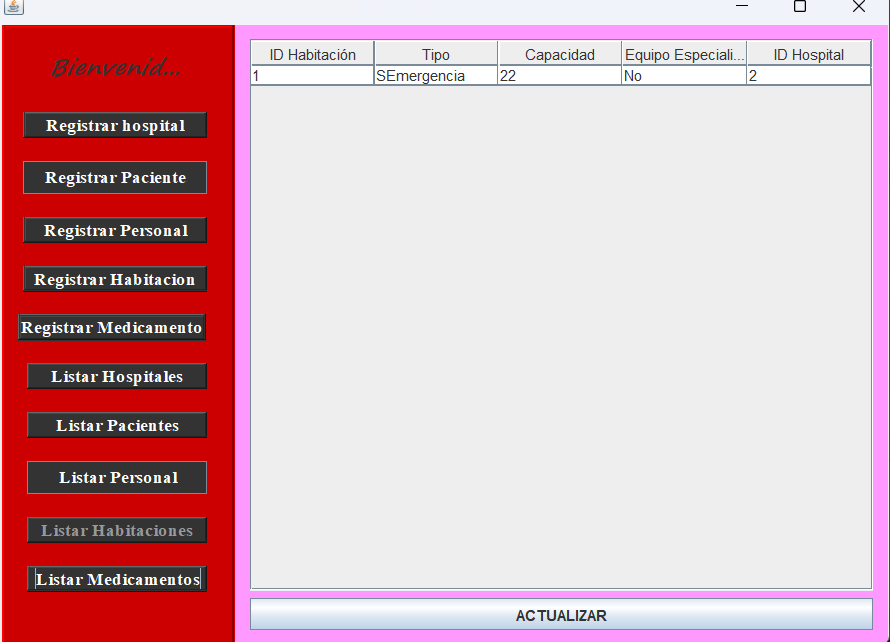
Listar Pacientes:



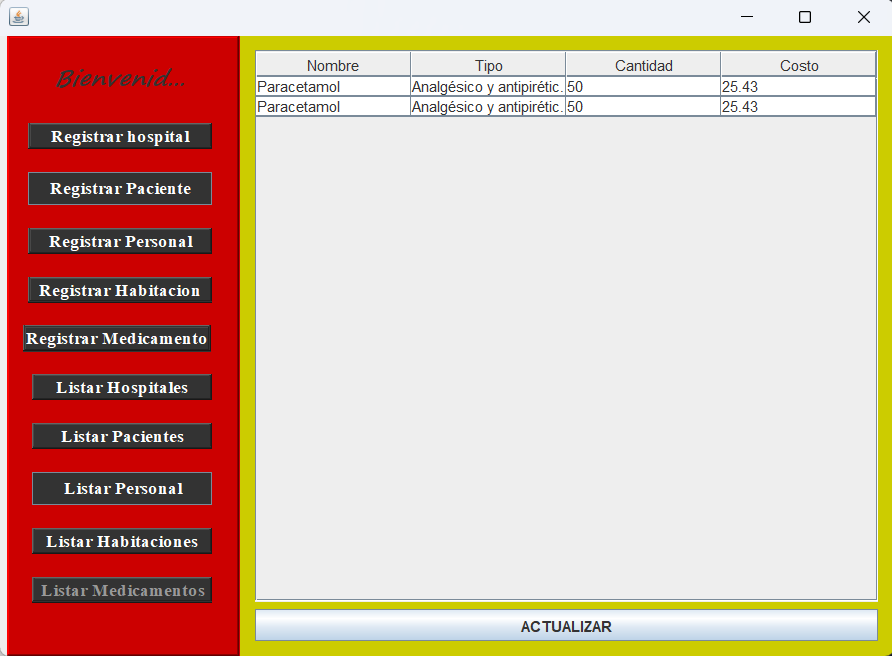
Listar Personal:



Listar Habitaciones:

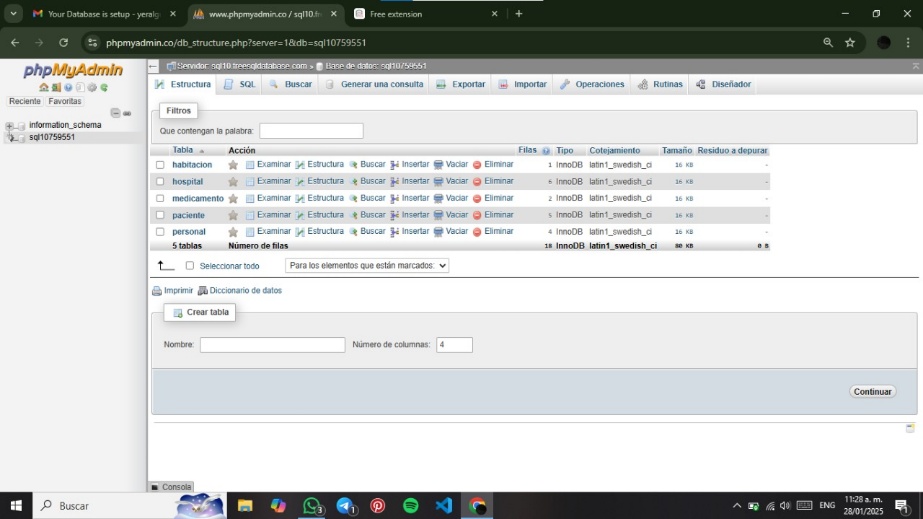


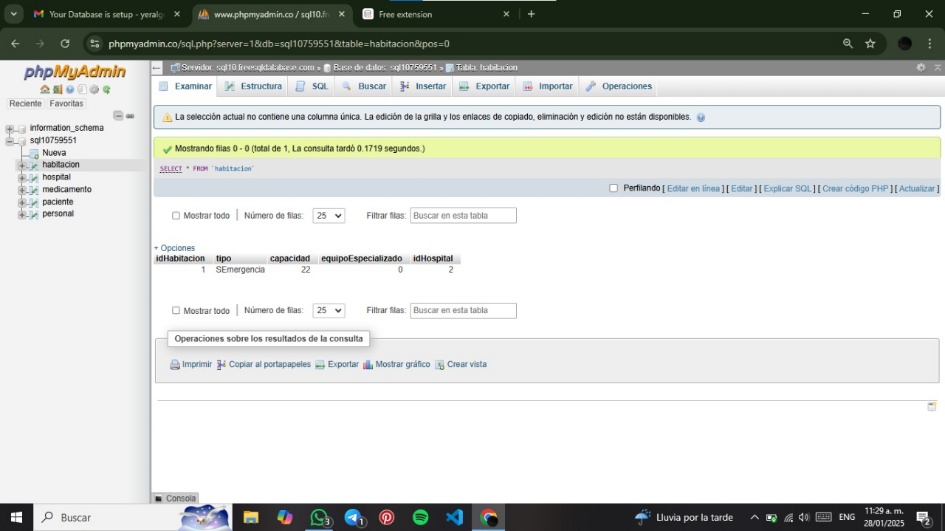
Listar Medicamentos:

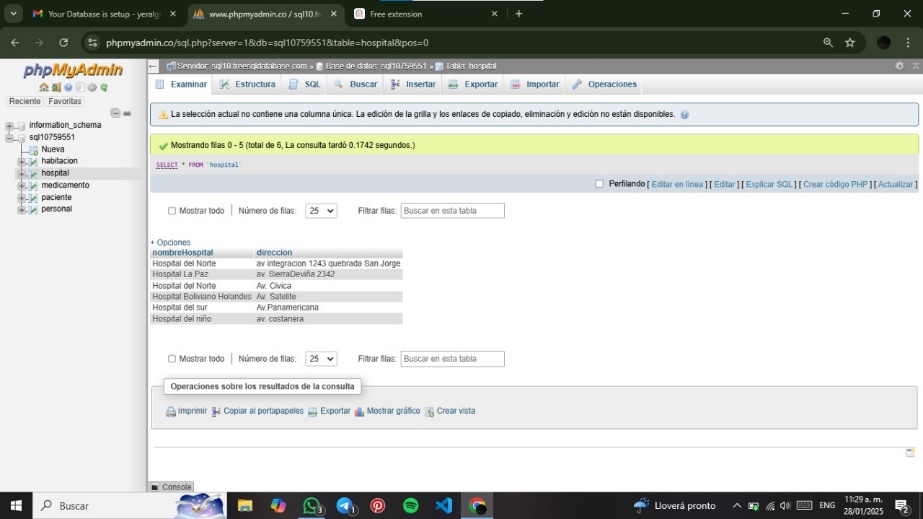


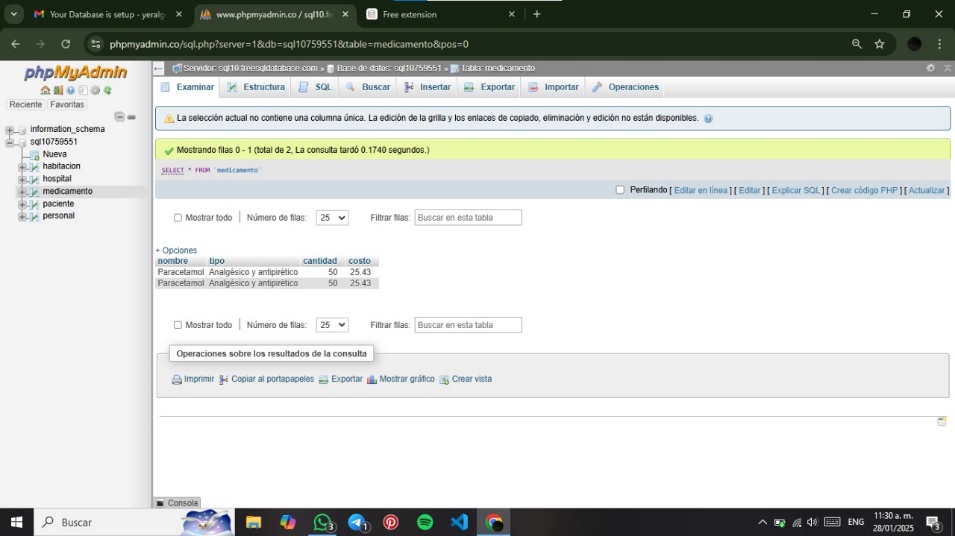
## 3.5 Manejo de Archivos

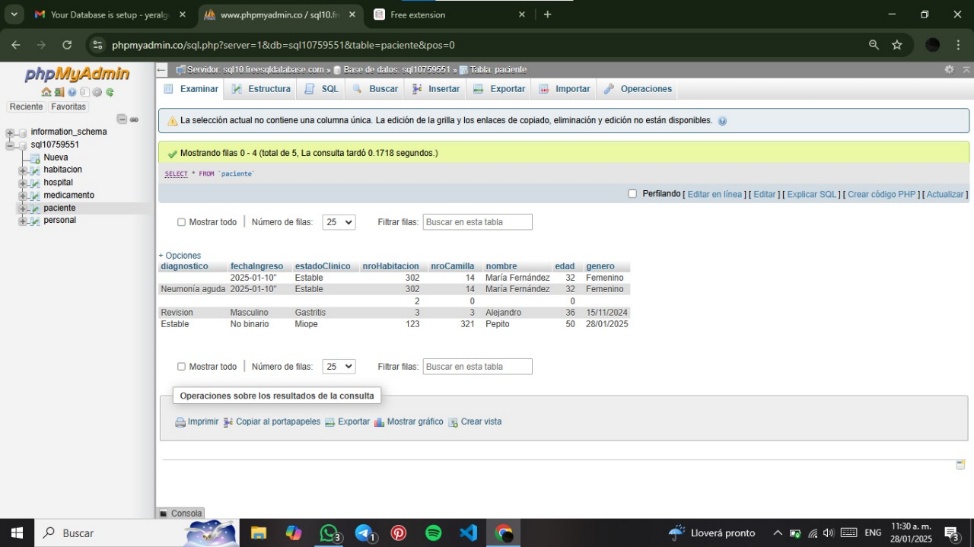
Se implementará la persistencia de datos utilizando conexiones a base de datos MySQL

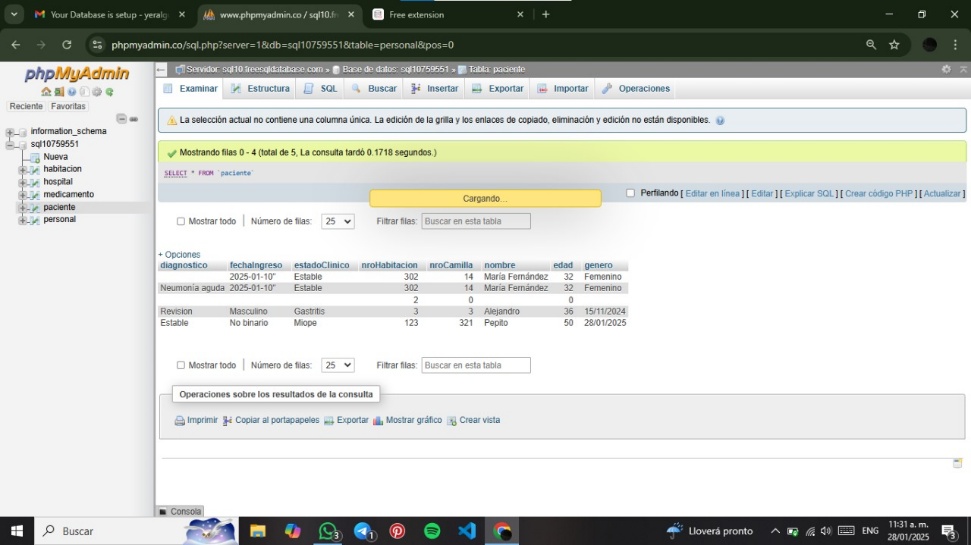


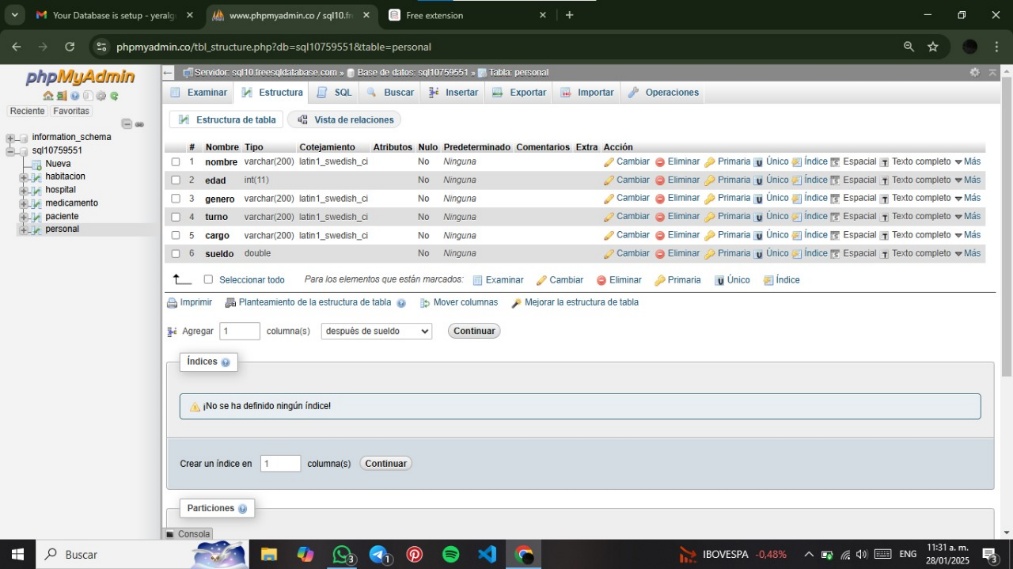












## 3.6 Uso de Patrones de Diseño

Para es te proyecto se usaron los siguientes patrones de diseño:

Text Field : para la entrada de los datos de tipo String.



Label: se utilizo los labels para títulos y subtitulos.



Spinner: Para la entrada de datos de tipo numéricos Enteros .



Botton: se usaron para ingresar, registrar, actualizar, y limpiar los datos.



Check Box: se utilizo este patron para los valores de tipo Boleano.



Eln la siguiente imagen podemos apreciar el uso de los elementos ya mencionados:



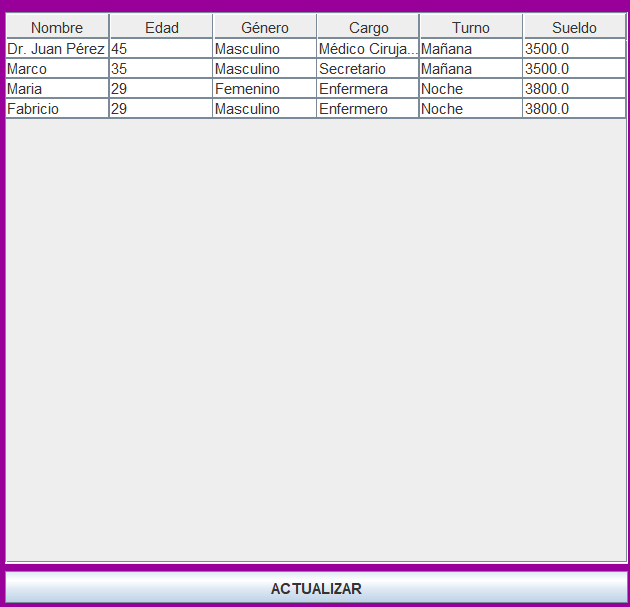
# 4. Pruebas del Sistema

Se realizarán pruebas en el sistema para verificar la correcta funcionalidad de los métodos principales.

Nos pide un a validación para que se ingresen todos los datos si no se ingresan todos los datos aparecerá una alerta:



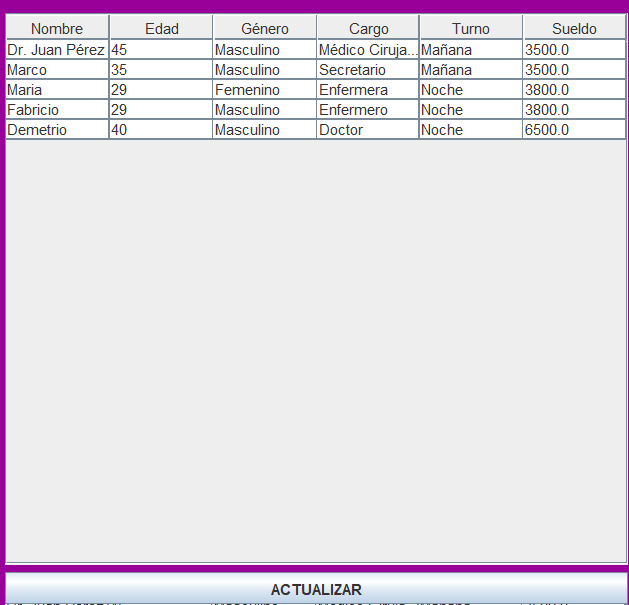
Como se puede observar en el sistema solo esta registrados 4 Empleados



Al registrar los datos del personal aparecerá un a confirmación si pudo completarse dicha acción:



Y podemos actualizar y verificar en la página que se haya registrado adecuadamente el personal personal,.



# 5. Conclusión

Este proyecto cumple con los aspectos importantes y comprendidos en la Programación Orientada a Objetos , De esta manera proporciona soluciones modulares, que pueden ser escalables para el Sistema de Gestión de un Hospital. A demás hace uso de los patrones de diseño de la interfaz para hacer el programa mas intuitivo con el usuario, haciendo que el sistema sea flexible y más fácil de utilizar.

.