PARCIAL 2

JHONATAN DAVID PEDRAZA PELAEZ - 1000258013

Trabajo presentado como requisito parcial de la asignatura de Programacion Orientada a Objetos al docente:

Ing. Nestor German Bolivar Pulgarin.

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERIA DEPARTAMENTO DE INGENIERÍA CIVIL Y AGRÍCOLA

BOGOTÁ D.C

2025

CONTENIDO

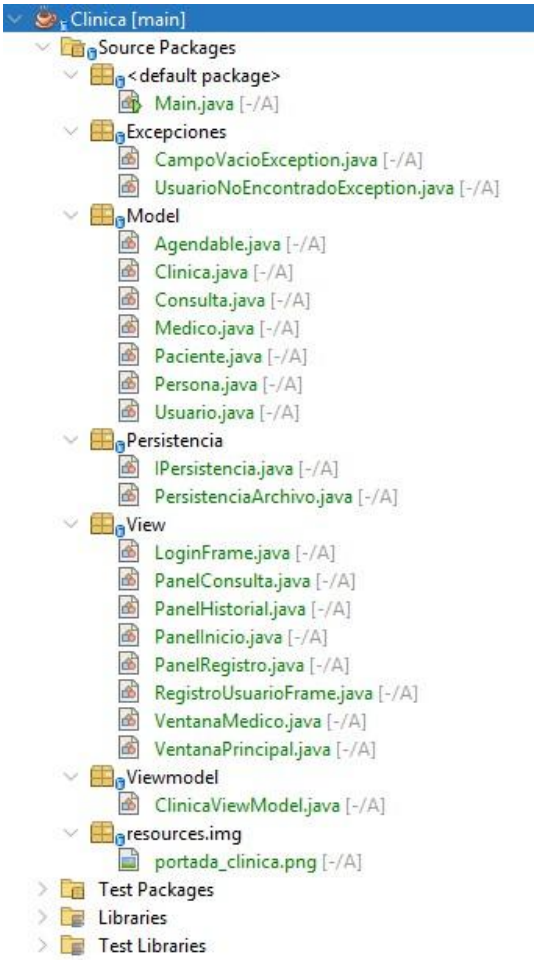
1. Introducción

El desarrollo de software orientado a objetos permite construir aplicaciones modulares, escalables y mantenibles, especialmente cuando se siguen patrones de diseño como **MVVM (Modelo-Vista-VistaModelo)**. Este informe presenta el desarrollo de una **aplicación de escritorio para la gestión de una clínica médica**, como parte del segundo parcial del curso de **Programación Orientada a Objetos**.

El principal objetivo de este trabajo es aplicar los principios fundamentales de la programación orientada a objetos —como encapsulamiento, herencia, polimorfismo y abstracción— mediante el diseño de un sistema que integre **interfaces gráficas con Java Swing**, persistencia de datos usando serialización de objetos, y un control de flujo que permita **interacción diferenciada según el rol del usuario** (administrativo o médico). Adicionalmente, se busca fomentar el pensamiento estructurado a través de la implementación del patrón MVVM, separando de forma clara la lógica de negocio, la lógica de presentación y la interfaz de usuario.

A través de este proyecto se fortalece la capacidad de diseñar arquitecturas limpias, reutilizables y visualmente amigables, acercando al estudiante a escenarios reales de desarrollo de software. La experiencia también incluye el uso de excepciones personalizadas, manipulación de archivos, manejo de colecciones y principios de diseño visual, brindando una visión integral del proceso de construcción de una aplicación robusta y funcional.

2. Arquitectura del proyecto (MVVM)



La arquitectura implementada en este sistema se basa en el patrón **MVVM (Modelo - Vista - VistaModelo)**, una variación del conocido patrón MVC, que promueve una separación clara entre la lógica de negocio, la lógica de presentación y la interfaz gráfica. Este enfoque permite mejorar la escalabilidad, reutilización de código y facilidad de mantenimiento.

Componentes principales:

1. Modelo (Model)

Representa las entidades del dominio:

- Paciente, Medico, Consulta

- Clase central: Clinica, que gestiona la colección de pacientes, médicos y consultas.
- Todas estas clases implementan Serializable para facilitar la persistencia.

2. VistaModelo (ViewModel)

Clase ClinicaViewModel actúa como puente entre la Vista y el Modelo:

- Recibe entradas desde la Vista
- Invoca métodos del Modelo (Clinica)
- Aplica lógica de validación y control de flujo
- Maneja excepciones personalizadas (CampoVacioException, UsuarioNoEncontradoException)

3. Vista (View)

Compuesta por interfaces gráficas desarrolladas en Java Swing:

- VentanaPrincipal, PanelRegistro, PanelConsulta, PanelHistorial, LoginFrame, PanelInicio, entre otros.
- Interactúan con el usuario y notifican a la ViewModel cuando se requiere ejecutar una acción.
- El diseño visual fue mejorado para tener un estilo profesional acorde al contexto clínico, utilizando GridBagLayout, BorderLayout y recursos visuales como imágenes e íconos.

Ventajas del MVVM en este proyecto:

- **Separación clara de responsabilidades**, evitando acoplamiento entre lógica y presentación.
- Facilita **la evolución del sistema**, permitiendo mejorar la interfaz sin afectar la lógica de negocio.
- Permite realizar **validaciones** y mostrar errores controlados al usuario desde la ViewModel.
- Mejora la **testabilidad** de la lógica de negocio.

3. Descripción funcional del sistema

La aplicación construida permite realizar la gestión integral de una clínica médica, diferenciando el comportamiento del sistema según el tipo de usuario que inicia sesión. Cada módulo fue diseñado para cumplir un propósito específico, asegurando una navegación intuitiva, con validaciones, persistencia y control de errores adecuado.

1 Login y creación de usuarios

Al iniciar el sistema, se presenta una ventana de **inicio de sesión** con los siguientes campos:

- Usuario
- Contraseña
- Tipo (admin o médico)

Los usuarios pueden ser registrados directamente desde esta misma ventana mediante el botón **"Crear Usuario"**, lo cual permite ampliar dinámicamente los accesos al sistema. Todos los usuarios son almacenados en el archivo usuarios.dat mediante serialización.

Además, el sistema valida:

- Usuario duplicado al registrar
- Contraseña incorrecta
- Tipo de usuario no correspondiente

Al iniciar sesión:

- Un **administrador** tiene acceso completo al sistema (registro, consultas e historial).
- Un **médico** tiene acceso únicamente al módulo de consultas.

2 Panel de Registro (Pacientes y Médicos)

Desde el menú principal, el usuario administrador puede acceder al **Panel de Registro**, que incluye dos pestañas:

- **Registrar Paciente**: requiere nombre, ID y correo electrónico.
- **Registrar Médico**: requiere nombre, ID, correo electrónico y especialidad.

Se aplican validaciones para evitar campos vacíos y se muestran mensajes de confirmación o error según el resultado.

Todos los registros se almacenan en memoria y se guardan en el archivo clinica.dat mediante serialización automática desde el ViewModel.

3 Panel de Consulta

Tanto los médicos como los administradores pueden acceder a este panel.

Permite:

- Seleccionar un **paciente** y un **médico** desde listas desplegables
- Ingresar los síntomas, diagnóstico y tratamiento
- Registrar la consulta

Se genera un **ID único automáticamente** para cada consulta.
El sistema valida que se haya seleccionado correctamente el paciente y el médico.

4 Panel de Historial Médico

Este módulo permite **consultar el historial de consultas**, usando dos pestañas:

- **Por paciente**: se ingresa el ID del paciente y se listan todas sus consultas médicas.
- **Por médico**: se ingresa el ID del médico y se listan todas las consultas que ha atendido.

En ambos casos, se utiliza un JTextArea para mostrar los resultados en un formato legible, y se maneja la excepción UsuarioNoEncontradoException cuando el ID ingresado no corresponde a un usuario registrado.

5 Persistencia de datos

La aplicación guarda y recupera automáticamente la información mediante serialización de objetos a archivos .dat, sin necesidad de bases de datos externas. Los archivos son:

- usuarios.dat → contiene todos los usuarios registrados.
- clinica.dat → contiene pacientes, médicos y consultas.

El sistema realiza lectura y escritura de estos archivos desde la ViewModel y desde los controladores de usuario, garantizando que los datos persisten entre ejecuciones.

4. Estilo visual y diseño

Uno de los aspectos más destacados de esta aplicación es el esfuerzo por presentar una **interfaz gráfica estética, profesional y amigable**, acorde al contexto de una clínica moderna. Se utilizó **Java Swing**, combinando diferentes gestores de diseño (BorderLayout, GridBagLayout, GridLayout) para lograr una distribución clara y balanceada de los elementos.

1 Paleta de colores y tipografía

La interfaz emplea una paleta de colores suave, basada en tonos:

- Celeste pálido (#F5FAFF, #D2EAFD)
- Blanco y gris claro para los fondos
- Azul vibrante para botones destacados

Esto transmite una sensación de limpieza, orden y confianza, típica de una clínica o centro de salud.

Se utilizó la fuente **SansSerif** en tamaño entre 14 y 18 pt, asegurando buena legibilidad en formularios, botones y etiquetas.

2 Panel de Inicio con imagen clínica

Al iniciar sesión, el usuario accede a una **pantalla de bienvenida con imagen de fondo**, acompañada de una frase institucional del estilo:

"Nuestras sonrisas hablan. Técnicas dentales en manos de profesionales."

Esta pantalla también incluye un botón de acceso rápido a **"Agendar cita"**, el cual redirige directamente al panel de consultas. La imagen utilizada simula el estilo moderno y publicitario de portales clínicos actuales.

3 Diseño responsivo y organizado

Cada panel (Registro, Consulta, Historial) está cuidadosamente estructurado para evitar saturación visual:

- Uso de EmptyBorder para dar espacios y márgenes
- Botones estilizados con colores suaves y letras destacadas
- Títulos centrados y paneles con pestañas para una navegación clara

4 Accesibilidad y usabilidad

- Se agregaron mensajes emergentes (JOptionPane) para **confirmar acciones exitosas o reportar errores**.
- Los campos de entrada son validados antes de registrar datos.
- Los formularios están organizados por tipo de usuario, permitiendo una **experiencia personalizada** según el rol (administrador o médico).

5. Capturas de pantalla

A continuación, se sugiere un conjunto de **capturas clave** que deberías incluir en tu informe (formato Word o PDF), junto con su **título y pie de figura** para facilitar la lectura del evaluador.

Figura 1. Ventana de Login

Vista inicial del sistema, donde el usuario puede ingresar con su nombre, contraseña y tipo de usuario. Incluye opción para crear nuevos usuarios.

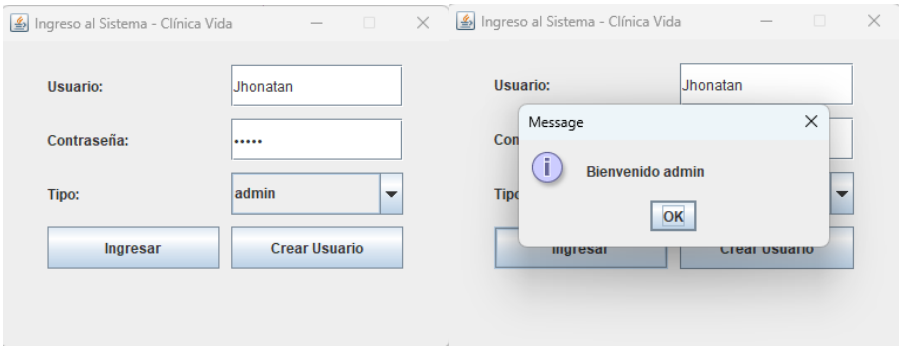


Figura 2. Registro de nuevo usuario

Ventana emergente que permite crear usuarios tipo médico o administrador. Los usuarios creados son almacenados en el archivo usuarios.dat.

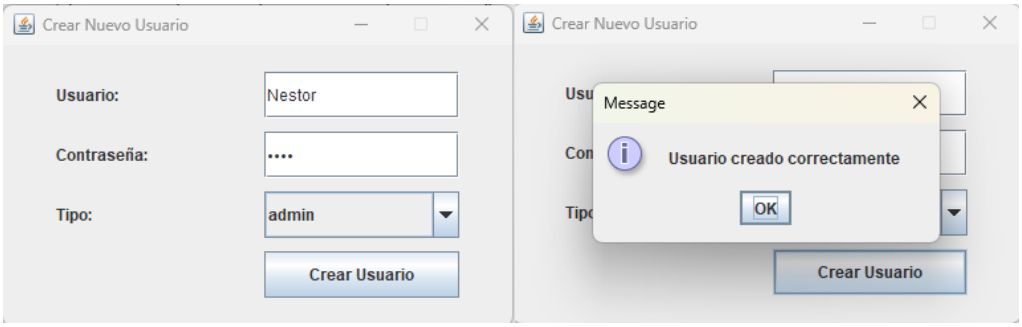


Figura 3. Interfaz de bienvenida

Pantalla principal con imagen de fondo y botón central de “Agendar Cita”. Diseño moderno, limpio y clínico.

Figura 4. Panel de Registro – Paciente

Formulario para registrar nuevos pacientes. Contiene validaciones y mensajes de éxito o error según los datos ingresados.

Figura 5. Panel de Registro – Médico

Formulario para registrar médicos, incluyendo campos como nombre, ID, correo y especialidad.

Figura 6. Panel de Consulta

Interfaz donde se selecciona un paciente y un médico, y se ingresan los síntomas, diagnóstico y tratamiento. Genera un ID único por consulta.

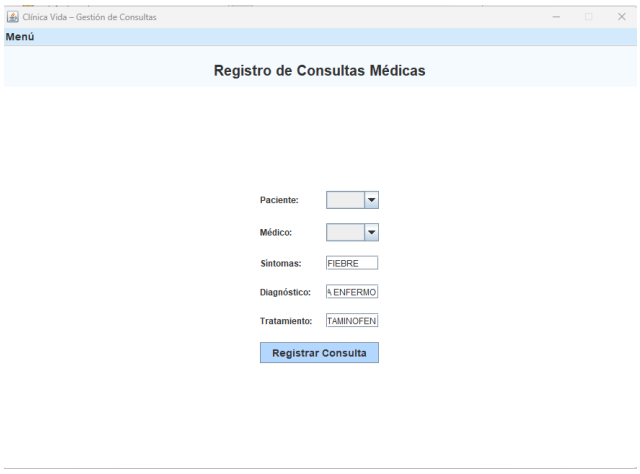


Figura 7. Panel de Historial – Por Paciente

Se muestra el historial médico completo de un paciente específico, consultando por su ID.

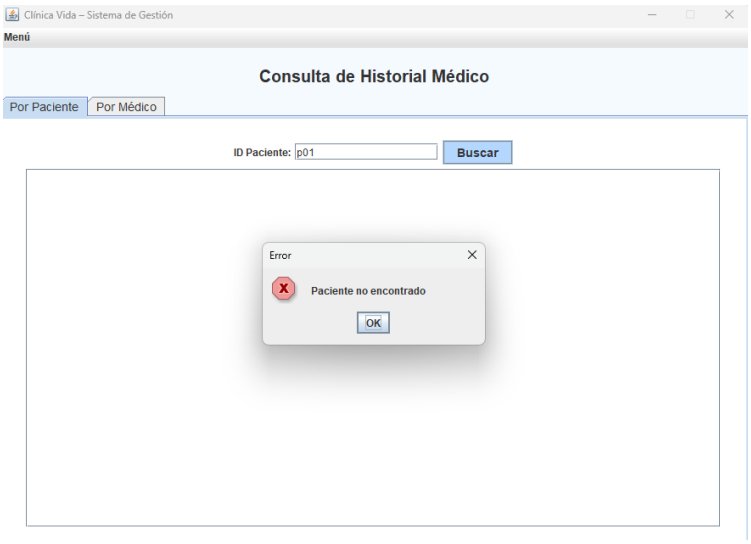


Figura 8. Panel de Historial – Por Médico

Muestra todas las consultas que ha realizado un médico específico, ingresando su ID.

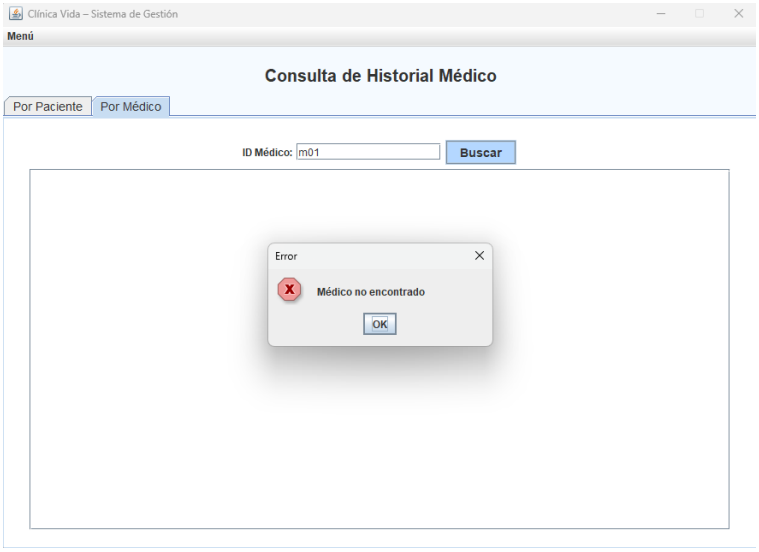


Figura 9. Estructura del proyecto en NetBeans

Organización del código fuente según el patrón MVVM: carpetas Model, View, Viewmodel, Persistencia, Excepciones, y recursos.

Figura 10. Archivos generados (.dat)

Vista del explorador de archivos con usuarios.dat y clinica.dat creados automáticamente por la aplicación.

6. Conclusiones

El desarrollo de esta aplicación permitió aplicar de manera práctica los conceptos fundamentales de la programación orientada a objetos, incluyendo herencia, polimorfismo, abstracción y encapsulamiento.

Se implementó el patrón MVVM para organizar el código de forma clara, lo que facilitó la separación entre la lógica de negocio y la interfaz gráfica. Además, se incorporaron funcionalidades de persistencia de datos, manejo de roles (médico y administrador) y validaciones con excepciones personalizadas.

El resultado es un sistema funcional, estable y visualmente amigable, que cumple con los objetivos del parcial y simula de forma efectiva el funcionamiento básico de una clínica médica digital.