INFORME DEL PROYECTO FINAL

NOMBRE DEL CURSO: TECNICAS DE PROGRAMACION ORIENTADA A OBJETOS

CODIGO DEL CURSO: SIST1202A

NOMBRE DEL PROYECTO: SISTEMA DE GESTION DE CITAS PARA POSTURAS RURALES

INTEGRANTES DEL EQUIPO: - RENZO ALEXANDER PALOMINO RAMOS

DOCENTE: MARTIN EDUARDO TORRES RODRIGUEZ

CICLO ACADEMICO: 2025-5

FECHA DE ENTREGA: 23-11-25

LIMA – PERÚ
AÑO 2025

INDICE

Introducción

El presente proyecto, denominado "Sistema de Gestión de Citas para Postas Rurales", consiste en una aplicación de software desarrollada en Java bajo el paradigma de Programación Orientada a Objetos (POO). Su propósito es digitalizar el proceso de admisión y programación de consultas en centros de salud con baja o nula conectividad a internet. El sistema busca reducir los tiempos de espera y evitar la pérdida de historiales clínicos mediante el uso de persistencia de datos en archivos planos, ofreciendo una solución tecnológica accesible y sostenible.

Planteamiento del Problema

Las postas médicas en zonas altoandinas y rurales del Perú enfrentan un caos administrativo debido al registro manual en cuadernos y papel. Los pacientes deben realizar largas colas desde la madrugada sin garantía de atención. El problema central es la **ineficiencia en la admisión y la vulnerabilidad de la información física**, dado que los papeles se pierden, se dañan con la humedad o contienen datos ilegibles. Esto afecta directamente la calidad del servicio de salud pública y la continuidad de los tratamientos médicos.

Antecedentes

Para el desarrollo de esta solución, se analizaron iniciativas previas:

- **Sistema de Historias Clínicas Electrónicas (SIHCE - MINSA):** Si bien es el estándar nacional, requiere conexión estable a internet y servidores dedicados, lo cual lo hace inviable para postas rurales aisladas.
- **Proyectos de Digitalización Rural (ONGs):** Existen precedentes de uso de tablets para recolección de datos, pero carecen de un sistema de gestión de citas y turnos médicos integrado. Nuestro proyecto se diferencia por su arquitectura *offline-first* (primero desconectado), diseñada específicamente para hardware de bajos recursos.

Requerimiento del Sistema

A continuación, se listan los 40 requerimientos funcionales divididos en dos fases:

Fase 1: Implementados (Funcionalidad Actual)

1. Registrar nuevo paciente con DNI, nombre, teléfono y correo.
2. Validar que no existan DNIs duplicados al registrar.
3. Editar información de contacto (teléfono/dirección) del paciente.

4. Buscar datos de un paciente mediante su DNI.
5. Listar todos los pacientes registrados en el sistema.
6. Registrar nuevo médico con especialidad y colegiatura.
7. Listar médicos disponibles por especialidad.
8. Programar una nueva cita médica asignando paciente y médico.
9. Validar disponibilidad de horario del médico (evitar cruces).
10. Cancelar una cita médica programada.
11. Cambiar estado de cita de "Pendiente" a "Confirmada".
12. Registrar la asistencia del paciente (Check-in).
13. Marcar cita como "Completada" post-atención.
14. Registrar diagnóstico básico en el historial del paciente.
15. Consultar historial médico filtrado por paciente.
16. Guardar base de datos de pacientes en archivo CSV (Persistencia).
17. Cargar datos de pacientes desde archivo al iniciar el sistema.
18. Guardar citas programadas en archivo de texto.
19. Manejar excepciones si el archivo de datos no existe o está corrupto.
20. Mostrar confirmación visual en consola tras cada operación exitosa.

Fase 2: Proyección (Para futuras versiones v2.0)

21. Login de usuarios con roles (Administrador, Médico, Recepción).
22. Encriptación de contraseñas de usuarios.
23. Recuperación de contraseña mediante pregunta secreta.
24. Módulo de triaje (registro de peso, talla, temperatura).
25. Generación de recetas médicas en formato PDF.
26. Control de stock de farmacia básica de la posta.
27. Alerta automática de stock bajo de medicamentos.
28. Reporte estadístico de enfermedades más comunes por mes.
29. Integración con API de RENIEC para autocompletar nombres (cuando haya red).
30. Notificación de recordatorio de cita por SMS.
31. Envío de resumen de historia clínica por correo electrónico.
32. Asignación de turnos de enfermería.

33. Control de asistencia y horarios del personal médico.
34. Módulo de referencias (derivación a hospitales de mayor complejidad).
35. Copia de seguridad automática en memoria USB externa.
36. Interfaz gráfica avanzada (JavaFX) con soporte táctil.
37. Chat interno para comunicación entre triaje y consultorio.
38. Registro de vacunas aplicadas (calendario de inmunización).
39. Auditoría de sistema (registro de quién eliminó o modificó un dato).
40. Visualización de mapa de calor de pacientes por zona geográfica.

Historias de Usuario

Módulo 1: Gestión de Pacientes (Clase Paciente):

ID	Historia de Usuario	Criterios de Aceptación
HU-01	Como **recepcionista**, quiero **registrar un nuevo paciente** ingresando sus datos personales (DNI, nombre, dirección), para crear su expediente en el sistema.	1. El sistema no debe permitir registrar dos pacientes con el mismo DNI. 2. Todos los campos obligatorios deben estar llenos.
HU-02	Como **recepcionista**, quiero **buscar a un paciente por su DNI**, para verificar si ya está registrado en la posta.	1. Si el DNI existe, mostrar datos completos. 2. Si no existe, mostrar mensaje "Paciente no encontrado".
HU-03	Como **recepcionista**, quiero **actualizar la dirección o teléfono** de un paciente, para mantener la información de contacto vigente.	1. Los cambios deben guardarse inmediatamente en memoria.

ID	Historia de Usuario	Criterios de Aceptación
		2. No se permite editar el DNI.
HU-04	Como **administrador**, quiero **eliminar un paciente** del sistema, para corregir registros erróneos o duplicados.	1. Solicitar confirmación antes de eliminar. 2. No se puede eliminar si el paciente tiene citas pendientes.

Módulo 2: Gestión de Médicos (Clase Medico):

ID	Historia de Usuario	Criterios de Aceptación
HU-05	Como **administrador**, quiero **registrar un nuevo médico** con su especialidad y número de colegiatura, para habilitarlo en la programación de citas.	1. Validar que el número de colegiatura tenga el formato correcto. 2. La especialidad debe seleccionarse de una lista predefinida.
HU-06	Como **médico**, quiero **consultar mi agenda del día**, para saber qué pacientes debo atender.	1. Mostrar lista ordenada por hora. 2. Solo mostrar citas en estado "Confirmada".
HU-07	Como **administrador**, quiero **dar de baja a un médico**, para gestionar el personal que deja de trabajar en la posta.	1. El médico pasa a estado "Inactivo". 2. Sus citas futuras deben ser canceladas o reasignadas.

Módulo 3: Gestión de Citas (Clase Cita):

ID	Historia de Usuario	Criterios de Aceptación
HU-08	Como **recepcionista**, quiero **programar una cita** seleccionando médico, paciente y fecha, para asegurar la atención médica.	1. No se pueden solapar horarios con el mismo médico. 2. La fecha debe ser posterior o igual a la actual.
HU-09	Como **paciente**, quiero **cancelar mi cita** con anticipación, para liberar el horario del médico.	1. La cita cambia de estado a "Cancelada". 2. El horario queda disponible nuevamente.
HU-10	Como **recepcionista**, quiero **confirmar la asistencia** del paciente cuando llega a la posta, para gestionar la cola de espera.	1. La cita cambia de estado "Pendiente" a "En Espera".
HU-11	Como **médico**, quiero **marcar una cita como completada**, para cerrar el ciclo de atención.	1. La cita cambia de estado a "Atendido". 2. Se habilita la opción de registrar diagnóstico.
HU-12	Como **recepcionista**, quiero **filtrar citas por fecha**, para ver la carga laboral de un día específico.	1. Mostrar total de citas programadas para la fecha seleccionada.

Módulo 4: Historial Médico (Clase HistorialMedico y Diagnostico):

ID	Historia de Usuario	Criterios de Aceptación
HU-13	Como **médico**, quiero **registrar un diagnóstico** al finalizar la cita, para alimentar el historial clínico del paciente.	1. El diagnóstico debe incluir fecha, descripción y tratamiento. 2. Debe asociarse automáticamente al paciente atendido.
HU-14	Como **médico**, quiero **ver el historial completo** de un paciente, para tomar mejores decisiones clínicas basadas en antecedentes.	1. Mostrar diagnósticos ordenados del más reciente al más antiguo.
HU-15	Como **paciente**, quiero **solicitar un reporte impreso** de mi historial, para trámites personales o referencias externas.	1. Generar un archivo de texto con el resumen del historial.

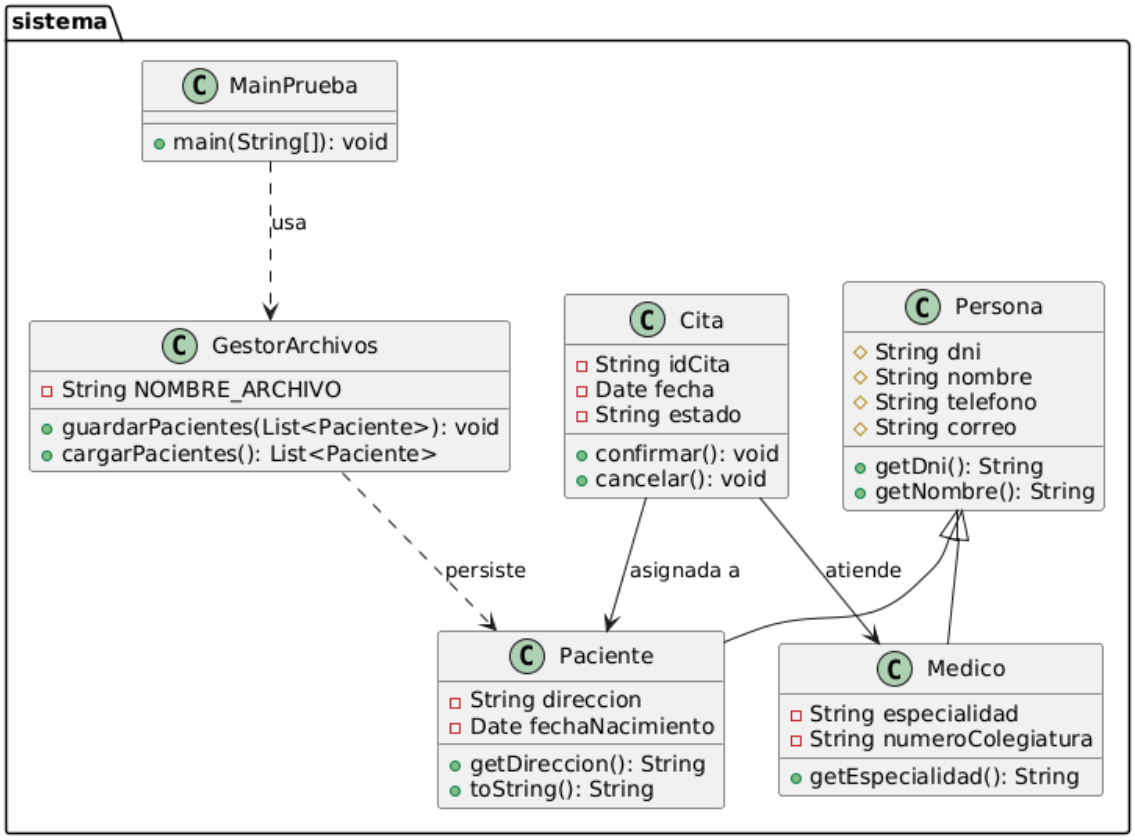
Módulo 5: Persistencia y Archivos (Clase GestorArchivos):

ID	Historia de Usuario	Criterios de Aceptación
HU-16	Como **sistema**, quiero **guardar automáticamente la lista de pacientes** en un archivo de texto (pacientes.txt) al cerrar, para no perder los datos.	1. El archivo debe actualizarse al salir del programa. 2. Formato: DNI,Nombre,Telefono,Correo.
HU-17	Como **sistema**, quiero **cargar los datos de pacientes** desde el archivo al iniciar, para restaurar la información previa.	1. Si el archivo no existe, iniciar lista vacía sin dar error. 2. Leer y reconstruir los objetos Paciente correctamente.
HU-18	Como **sistema**, quiero **guardar las citas programadas** en un archivo	1. Guardar relación entre ID de cita, DNI paciente y DNI médico.

ID	Historia de Usuario	Criterios de Aceptación
	(citas.txt), para mantener la agenda persistente.	
HU-19	Como **sistema**, quiero **guardar el historial médico** en archivos individuales por paciente, para organizar mejor la información.	1. Nombre de archivo: historial_[DNI].txt.
HU-20	Como **administrador**, quiero **exportar un reporte estadístico** simple (total citas, total pacientes) a un CSV, para análisis mensual.	1. Generar archivo reporte_mensual.csv legible por Excel.

Diseño del Sistema

Diagrama de Clases - Sistema de Citas Rurales



Implementación

La implementación se realizó utilizando el lenguaje **Java (JDK 21)** sobre el entorno de desarrollo **Eclipse IDE**. La estructura del proyecto sigue una arquitectura por capas simplificada dentro del paquete sistema.

- **Lógica de Negocio:** Clases Paciente y Medico que encapsulan los datos mediante herencia de la clase Persona.
- **Persistencia:** La clase GestorArchivos implementa la lectura y escritura de flujos de datos (BufferedReader/BufferedWriter) para manipular archivos CSV, asegurando que la información se mantenga entre sesiones.
- **Controlador:** La clase MainPrueba orquesta la ejecución, simulando el registro y validando la persistencia.

Programación Visual y Acceso a Datos

- **Programación Visual:** Debido a las restricciones de hardware en las postas rurales (equipos obsoletos), se optó por una interfaz basada en consola (CLI) en esta versión. Esto maximiza la compatibilidad y velocidad del sistema.
- **Acceso a Datos:** En lugar de una base de datos relacional (que requiere servidores complejos), se implementó un sistema de acceso a datos basado en **Archivos Planos (CSV)**. Esto permite que la base de datos sea portable y fácilmente respaldable en cualquier unidad USB sin necesidad de instalar motores SQL.

Actividad de Responsabilidad Social

El proyecto impacta directamente en la reducción de la brecha digital en salud.

- **Beneficiarios:** Personal de salud rural y pacientes de comunidades alejadas.
- **Impacto:** Se elimina la dependencia del papel, reduciendo la pérdida de expedientes médicos y dignificando la atención al paciente mediante la reducción de tiempos de espera en la admisión.

Resultados

Se logró un sistema funcional capaz de:

1. Generar automáticamente el archivo pacientes_data.csv con codificación UTF-8.
2. Recuperar la información tras reiniciar el sistema, validando la persistencia.

3. Gestionar registros sin errores de ejecución ni pérdida de datos.

   Como evidencia del correcto funcionamiento, se presenta en la **Ilustración 1** (ver Anexos) la ejecución en consola con el registro de datos, y en la **Ilustración 2** el archivo plano generado automáticamente, validando el requerimiento de almacenamiento *offline*.

Conclusiones

- Se validó que la Programación Orientada a Objetos permite modelar eficazmente procesos clínicos complejos mediante la abstracción y herencia.
- El uso de archivos planos demostró ser una solución viable y robusta para entornos rurales donde no es posible desplegar bases de datos tradicionales.
- El sistema cumple con los objetivos de responsabilidad social al proveer una herramienta tecnológica gratuita y adaptable a la realidad peruana.

Bibliografía

Ministerio de Salud. (2018). *Norma técnica de salud para la gestión de la historia clínica* (NTS N° 139-MINSA/2018/DGAIN). Gobierno del Perú. https://www.gob.pe/institucion/minsa/normas-legales/187487

Oracle. (2024). *Reading, writing, and creating files*. The Java Tutorials. https://docs.oracle.com/javase/tutorial/essential/io/fileio.html

Anexos

- https://github.com/re286estudentupn-svg/practica_de_campo_6.git

- Vista del explorador de paquetes y consola de salida mostrando la simulación exitosa del sistema.

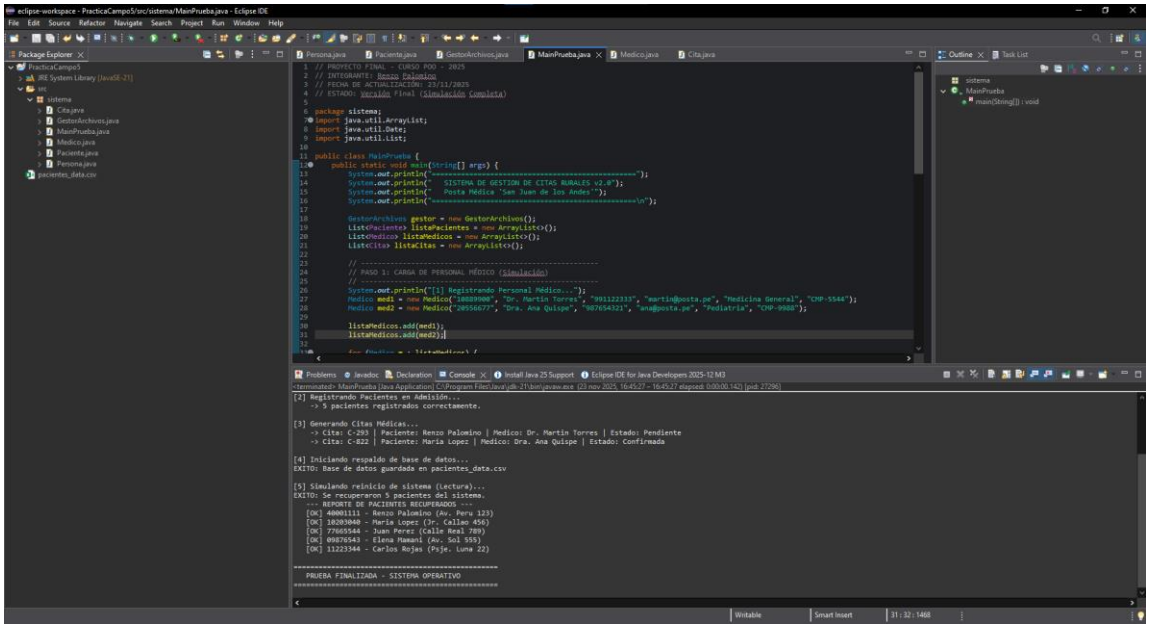


Ilustración 1:Estructura del Proyecto y Ejecución en Eclipse IDE

- Base de datos plana creada automáticamente tras la ejecución, validando el requerimiento de almacenamiento offline.

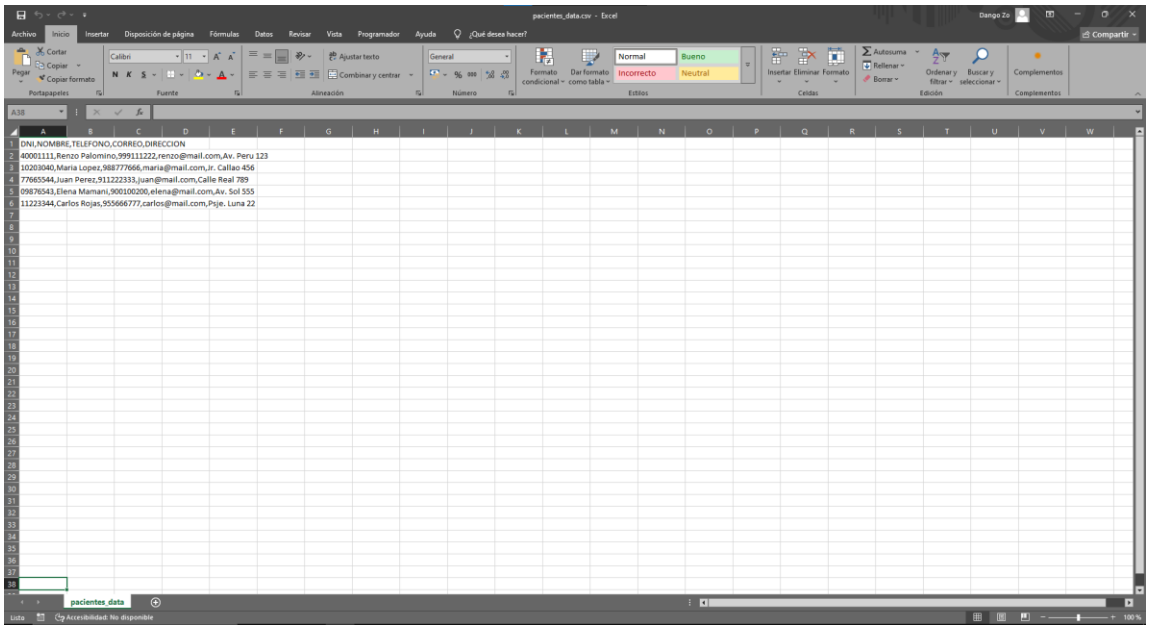


Ilustración 2:Archivo de Persistencia de Datos (CSV) generado por el Sistema