Sistema de Gestión Farmacéutica

Liberto Vázquez, Gaizka Yustes y Ander Lecue
Programación 3
Proyecto-Prog.III-9
16 de enero de 2026

ÍNDICE

1.-Valoración del resultado final del proyecto

1.1. Cumplimiento de las expectativas iniciales

En términos generales, el producto final se alinea de forma satisfactoria con los objetivos establecidos al inicio del cuatrimestre. El propósito principal era desarrollar una aplicación de escritorio en Java puro destinada a la gestión operativa de una red de farmacias, y el resultado entregado cumple efectivamente con esta premisa fundamental.

Sin embargo, más allá del cumplimiento básico, el desarrollo ha superado las expectativas iniciales en cuanto a funcionalidad. La conexión a la base de datos mediante JDBC se comporta de manera sólida, preservando la integridad referencial incluso tras la inserción masiva de datos de prueba. Visualmente, aunque Swing impone ciertas limitaciones estéticas frente a tecnologías más actuales, la interfaz resultante es coherente, funcional y está libre de errores graves de visualización, lo que confirma la adecuación de las decisiones de diseño adoptadas. Asimismo, la aplicación es lo más estético posible dentro de que la mayoría de ventanas gráficas son tablas de datos y puede llegar a dar la sensación de una interfaz clásica.

1.2. Implementación de la funcionalidad

Podemos confirmar que se ha alcanzado el 100 % de la funcionalidad obligatoria establecida en la especificación de requisitos. La aplicación cubre de forma completa los módulos principales:
Seguridad y Acceso: El sistema de Login y Registro funciona correctamente, discriminando usuarios y gestionando sesiones de manera segura contra la base de datos.
Gestión de Inventario (Almacén): Se permite la visualización, filtrado y modificación de productos farmacéuticos.
Gestión Comercial (Pedidos): Se ha implementado con éxito tanto el historial de pedidos pasados como la generación de nuevos pedidos, donde se aplican los algoritmos de recursividad exigidos para el cálculo de impuestos y totales anidados.
Gestión de clientes: Se han asignado unos clientes en específico a las diferentes sedes de la farmacia, recogiendo información del cliente y de las compras realizadas por el mismo. Pudiendo además añadir, eliminar y modificar nuevos clientes a las diferentes sucursales.
Gestión de trabajadores: Se han asignado unos trabajadores a las diferentes sedes de la farmacia, recogiendo información del trabajador. Pudiendo además añadir, eliminar y modificar nuevos clientes a las diferentes sucursales.
Gestión de compras: Se han asignado unas compras realizadas por los diferentes clientes en las diferentes sucursales.
Gestión de ventas: Se ha implementado la gestión de las compras de los clientes pero con la visión de la farmacia, siendo estas las ventas realizadas por las diferentes sucursales a los diferentes clientes.
Requisitos Técnicos Avanzados: Se ha cumplido estrictamente con la implementación de Multithreading (hilos) para los relojes en tiempo real y las pantallas de carga.

1.3. Mejoras y valor añadido respecto a la idea inicial

Durante el desarrollo surgieron oportunidades para enriquecer la aplicación más allá de lo estrictamente exigido. Las principales mejoras incorporadas son:

Sistema de Datos Sintéticos Masivos: Inicialmente, se planeaba probar la aplicación con una docena de productos manuales. Sin embargo, mediante el uso de Inteligencia Artificial Generativa y un programa en Java realizado por nosotros, creamos scripts para poblar la base de datos con miles de registros (productos, clientes, trabajadores, compras,etc.). Esto aumentó el realismo de las demostraciones y obligó a optimizar consultas para entornos de alto volumen.

Dashboard con Alertas Proactivas: En lugar de un menú estático, desarrollamos un panel de bienvenida inteligente. Antes de acceder a la gestión de una farmacia concreta, el sistema consulta en segundo plano y muestra información relevante (pedidos pendientes, productos en stock crítico, etc.), mejorando notablemente la usabilidad.

El éxito alcanzado en el proyecto se explica principalmente por tres decisiones de enfoque que marcaron todo el proceso de desarrollo.

En primer lugar, optamos por un desarrollo progresivo. Comenzamos creando la interfaz gráfica con Swing, lo que nos permitió definir desde muy pronto los flujos de usuario y la disposición general de las pantallas. Con ese armazón visual ya claro, pasamos a implementar la lógica de negocio, enfrentándonos en esa etapa central a los aspectos más técnicos: concurrencia con hilos y algoritmos recursivos. Solo en la fase final conectamos la persistencia, integrando la base de datos con una lógica que ya funcionaba correctamente. Este orden nos permitió trabajar siempre sobre una aplicación que se podía ejecutar y ver, lo que facilitó mucho comprender en cada momento el estado real del avance y detectar a tiempo posibles desviaciones.

Asimismo, la gestión del equipo fue determinante para mantener este ritmo de trabajo. Lejos de asignar roles estancos, se optó por una colaboración horizontal y transversal, donde todos los integrantes participaron activamente en todas las áreas del desarrollo, desde el diseño de las ventanas hasta las consultas SQL. Esta ausencia de roles fijos fomentó una propiedad colectiva del código, impidiendo la formación de "silos de conocimiento". Al conocer todos los miembros el funcionamiento integral del sistema, la integración de la base de datos en la etapa final fue mucho más fluida, ya que cualquier integrante tenía el contexto necesario para adaptar la interfaz y la lógica a los requisitos de persistencia.

Finalmente un segundo factor clave fue la organización del equipo. En vez de repartir tareas con roles cerrados y compartimentos estancos, elegimos una colaboración mucho más horizontal: todos participamos activamente en prácticamente todas las capas del sistema, desde el diseño de formularios hasta la redacción de consultas SQL. Al tener todos una visión bastante completa del sistema, la integración final de la base de datos resultó notablemente más sencilla y natural, ya que cualquier persona del equipo disponía del contexto necesario para realizar los ajustes requeridos en la interfaz o en la lógica de negocio.

2.-Valoración del proceso de realización del proyecto

La coordinación del equipo se basó en un modelo de colaboración equitativa, no obstante cada uno de los participantes ha desarrollado y se ha centrado en unas clases más que en otras, repartiendo así el trabajo a realizar. Todos los integrantes han contribuido con una carga de trabajo similar, participando activamente tanto en la codificación como en la documentación. Apenas han habido desacuerdos en el grupo, como mucho aquellas discrepancias surgidas fueron exclusivamente técnicos ha la hora de decidir la funcionalidad de la aplicación (arquitectura de clases, implementación de recursividad, etc.) y se resolvieron mediante debate y consenso, priorizando siempre la solución más adecuada para el proyecto.

El seguimiento del esfuerzo se realizó mediante la hoja de planificación compartida y las líneas de código codificadas en la aplicación y medidas en base a github analyzer2. El cómputo final arroja aproximadamente 55 horas reales por persona, frente a las 60 inicialmente estimadas. El decremento se atribuye principalmente al factor de que en un principio teníamos pensado realizar una aplicación con mayor cantidad de JFrames, sin embargo tras la primera revisión nos diste a entender que no era necesario tal cantidad de ventanas gráficas por lo que las horas empleadas para la realización del codigo, finalmente no ha sido necesario emplearlas.

El trabajo no se distribuyó de forma lineal a lo largo del semestre. Identificamos un patrón de intensidad variable: se observa un inicio intenso en el diseño de la interfaz, una etapa intermedia de menor ritmo coincidente con exámenes parciales de otras asignaturas, y un sprint final muy concentrado durante las últimas tres semanas, dedicado a la integración de la base de datos y la depuración exhaustiva. Además, se puede observar la variabilidad de intensidad medido también en base a las fechas de entrega de las partes correspondientes del proyecto, haciendo que las semanas cercanas a la fecha de entrega sean más intensas.

El uso de GitHub fue esencial, aunque presentó dificultades iniciales con conflictos de fusión al modificar simultáneamente archivos de la interfaz lo que nos enseñó la importancia de una comunicación previa antes de realizar commits y pushes. Estas incidencias se redujeron notablemente al mejorar la comunicación previa y la segmentación de tareas.

3. Aspectos Positivos y Negativos del Proceso

Aspectos Positivos

- **Todos hicimos de todo**: Una de las mejores decisiones fue no repartirnos el trabajo en islas aisladas. Al participar todos en el desarrollo de la base de datos, la lógica y la vista, conseguimos que nadie se sintiera perdido. El aprendizaje fue mucho más justo y todos acabamos entendiendo el proyecto al completo.
- **La motivación de "ver" el proyecto**: Empezar por la interfaz gráfica fue un acierto total para el ánimo del equipo. Ver ventanas que funcionaban desde el principio nos motivó mucho más que ver solo líneas de código en la consola, y además nos sirvió para pillar fallos de usabilidad muy pronto.
- **Superar el reto de Swing**: Sabemos que la concurrencia es difícil, y por eso estamos orgullosos de cómo la manejamos. Conseguir que la aplicación no se quedara "congelada" o bloqueada mientras cargaba datos fue un logro técnico importante para nosotros.

Aspectos Negativos

- **El error de dejar la base de datos para el final**: Siendo sinceros, aquí nos equivocamos. Al postergar la persistencia, cuando quisimos conectarla tuvimos que reescribir (refactorizar) un montón de código que ya dábamos por terminado. La lección nos ha quedado clarísima: el modelo de datos se define al principio, no al final.
- **Los conflictos en GitHub**: Al principio sufrimos bastante pisándonos el código unos a otros porque trabajamos todos sobre la misma rama. Aprendimos "a la fuerza" que usar un sistema de ramas por funcionalidad (feature branches) no es algo opcional, sino obligatorio para no perder el tiempo arreglando conflictos.
- **Diseñar "a ojo" consume mucho tiempo**: Subestimamos lo que tardas en cuadrar una ventana en Swing picando código directamente. Perdimos muchas horas de prueba y error que nos habríamos ahorrado si hubiéramos hecho un boceto rápido en papel o un prototipo antes de ponernos a programar.

4.- Lecciones aprendidas

La experiencia acumulada nos permite una reflexión constructiva sobre cómo abordaríamos el desarrollo de software en el futuro. Si tuviésemos que iniciar el proyecto nuevamente, aplicaríamos cambios estructurales clave:

- Prioridad al modelo de datos: Aunque comenzar por la interfaz gráfica fue motivador, la falta de una base sólida nos obligó a refactorizaciones costosas. La lección más crítica es definir firmemente el diagrama Entidad-Relación y las estructuras de datos antes de avanzar masivamente en la codificación de la vista.

- Control de versiones riguroso: Adoptaríamos desde el primer día un sistema de ramas por funcionalidad (feature branches). Esto es vital para aislar los cambios y evitar los conflictos de código que experimentamos al trabajar simultáneamente sobre los mismos archivos.

- Prototipado previo: Incluiríamos una fase de prototipado visual antes de codificar interfaces complejas para clarificar requisitos sin escribir código prematuro.

Evaluación del apoyo docente
Consideramos que los recursos y la supervisión proporcionados han sido plenamente suficientes y adecuados. No estimamos necesaria la inclusión de tutorías adicionales o material extra, dado que la metodología de seguimiento continuo cubrió nuestras necesidades.

El sistema de entregas y revisiones periódicas actuó como un mecanismo de control de calidad efectivo. Este feedback continuo nos permitió conocer los aciertos, identificar errores técnicos y reconducir el proyecto oportunamente sin necesidad de intervenciones externas.

5.- Justificación de la distribución de esfuerzos en GitHub

Tras contrastar los resultados de la herramienta github-analyzer con la realidad de nuestro flujo de trabajo, concluimos que el esfuerzo ha sido equitativo, independientemente de las discrepancias cuantitativas que puedan aparecer en las métricas de aportación de código.

Es fundamental interpretar estos datos con cautela: el número de commits o líneas de código no refleja necesariamente la complejidad intelectual ni la dedicación invertida en tareas críticas que no siempre generan código directo, como el diseño de la arquitectura, la depuración compleja, la documentación o la coordinación del equipo.

Dado que la responsabilidad, la carga de trabajo y la resolución de problemas fueron compartidas de manera solidaria durante los tres meses de desarrollo, consideramos justa una calificación idéntica para todo el equipo. En consecuencia, sobre una valoración total de 100 puntos, realizamos una distribución equitativa de 33,3 puntos por integrante, reflejando la paridad en el compromiso y la ejecución técnica demostrada

6.- Uso de herramientas de Inteligencia Artificial Generativa

Durante el proyecto se emplearon de forma complementaria herramientas como ChatGPT, Gemini, Claude y GitHub Copilot.
Su contribución más valiosa se dio en:

- Generación de conjuntos masivos de datos sintéticos realistas
    - Facilitaron la creación masiva de ficheros CSV con miles de registros de prueba, alcanzando un nivel de detalle y realismo inviable de generar manualmente.
- Ayuda en la interpretación y resolución de errores técnicos.
    - Funcionaron como asistentes para la interpretación de errores y la comprensión de la lógica subyacente, acelerando el ciclo de corrección y aprendizaje.
- Creación y adaptación de código repetitivo o boilerplate.
    - Agilizaron la escritura de estructuras repetitivas (boilerplate), constructores y la adaptación de patrones arquitectónicos específicos.

Estas herramientas, utilizadas con criterio, permitieron liberar tiempo para centrarse en aspectos arquitectónicos y de diseño más relevantes. Se reconoce, no obstante, la importancia de comprender en profundidad las soluciones obtenidas y no limitarse a copiarlas, para garantizar un aprendizaje real.
De cara al futuro profesional, se percibe a la inteligencia artificial como un asistente de gran valor que transformará el rol del desarrollador hacía funciones de mayor nivel: arquitectura, validación y auditoría de software.