Seminario de Practica Licenciatura en Informática

Clínica Horizonte

Alumno: Lucero Gauna, Joaquin Claudio

Legajo: VINF015929

Materia: Seminario de Practica de Informática

Profesor: Aranda, Marcos Dario

Virgolini, Pablo Alejandro

Fecha: 16/11/2025

Contenido

Titulo

Desarrollo de un sistema de gestión y seguimiento del stock de insumos médicos en la Farmacia Central de la clínica, permitiendo asegurar la continuidad asistencial y optimizar el uso de recursos

Introducción

La Clínica Horizonte, con más de 40 años de trayectoria en la ciudad de Córdoba, se ha consolidado como institución referente en la atención médica integral, brindando servicios de guardia, internación, diagnóstico y cirugías. Dentro de su estructura, la Farmacia Central constituye un área clave, dado que es la encargada de proveer insumos médicos a todos los sectores de la organización.

Actualmente, la clínica enfrenta dificultades en la gestión de dichos insumos, ya que el sistema administrativo en uso se orienta principalmente a la facturación y gestión de pacientes, sin cubrir las necesidades de control detallado de stock. Esto ocasiona inconvenientes que repercuten directamente en la calidad de la atención.

Con este proyecto, se busca diseñar y desarrollar un sistema de gestión de stock de insumos médicos que permita registrar ingresos y egresos de manera informatizada, controlar niveles de stock mínimos, emitir alertas automáticas y generar reportes. De esta forma, se contribuirá a optimizar el uso de recursos y a garantizar la disponibilidad de insumos esenciales para la atención médica.

Antecedentes

En la actualidad, la clínica dispone de un software administrativo que solo gestiona compras y facturación, pero no incluye un módulo específico para la administración de insumos médicos. Por esta razón, gran parte del control se realiza de forma manual mediante planillas de cálculo o registros en papel.

Este método genera problemas recurrentes, entre ellos:

- Desabastecimiento imprevisto de insumos críticos como guantes, jeringas o soluciones fisiológicas.
- Demoras en los procedimientos médicos, al tener que esperar reposición.
- Incremento de costos, debido a compras de urgencia y poca planificación.
- Falta de trazabilidad, ya que no existe registro claro de qué insumos se entregan, a quién y en qué momento.

La ausencia de un sistema integrado que unifique la gestión de insumos conlleva una serie de riesgos tanto operativos como económicos, que impactan en la continuidad de la atención y en la eficiencia de la institución.

Descripción del área problemática

La problemática se centra en la Farmacia Central, que recibe, almacena y distribuye insumos médicos al resto de la clínica. Actualmente, no se dispone de una herramienta tecnológica que brinde información confiable y en tiempo real sobre el stock.

Los principales problemas detectados son:

- Control manual de ingresos y egresos.
- Falta de visibilidad del stock real disponible.

- Ausencia de alertas de punto de pedido crítico.
- Procesos de reposición basados en comunicación informal.

Estos factores generan ineficiencias en la operación y ponen en riesgo la atención médica en momentos críticos, como cirugías o emergencias.

Justificación

La implementación de un sistema de gestión de stock en la Farmacia Central permitirá:

- Asegurar la disponibilidad de insumos médicos fundamentales, evitando la interrupción de servicios.
- Optimizar el uso de recursos, disminuyendo costos por compras de urgencia y pérdidas por vencimiento.
- Mejorar la trazabilidad de movimientos, registrando ingresos y egresos en forma ordenada.
- Fortalecer la planificación y la toma de decisiones, mediante reportes que reflejen el consumo y el estado de los insumos.

En conclusión, el proyecto se justifica porque aborda un problema real y recurrente de la clínica, que afecta tanto la eficiencia operativa como la calidad del servicio asistencial.

Objetivo general del proyecto

Implementar en la Clínica Horizonte un proceso integral y estandarizado para la gestión de insumos médicos en Farmacia Central, que asegure continuidad asistencial, reduzca quiebres de stock y compras de urgencia, y provea trazabilidad de ingresos y egresos con información confiable para la toma de decisiones.

Objetivo general del sistema

Gestionar los movimientos de insumos médicos registrando toda la información pertinente de las transacciones realizadas —ingresos por compras y egresos hacia los servicios de la clínica—, y a la vez administrar la información del personal mediante la creación y mantenimiento de perfiles de usuario (administrador y auxiliar de farmacia). En síntesis, centralizar en un único repositorio los datos de stock, movimientos y usuarios de la Farmacia Central, ofreciendo información consistente, confiable y oportuna para la toma de decisiones y asegurando la continuidad asistencial.

Limites

El sistema entra en acción desde que los insumos médicos llegan a la Farmacia Central y son registrados como ingreso de stock (por compras aprobadas), y finaliza cuando dichos insumos se egresan y quedan asentados como entregados a los servicios de la clínica (guardia, internación, quirófano o consultorios), actualizando las existencias y, cuando corresponda, disparando alertas por punto de pedido. El ámbito se restringe a un único depósito (Farmacia Central) y a la administración de usuarios (administrador y auxiliar de farmacia) necesarios para ejecutar estas operaciones.

Alcance

Los procesos que serán soportados dentro del sistema (ámbito: único depósito – Farmacia Central) son:

- Registro de ingresos de insumos (compras) y actualización en línea del inventario.
- Registro de egresos de insumos hacia los servicios de la clínica (guardia, internación, quirófano, consultorios) con descuento automático de stock.
- Verificación de inventario: consulta por producto, visualización de stock mínimo y listado de críticos (stock ≤ mínimo) y próximos a vencer (si se carga vencimiento).
- Gestión de usuarios: creación, edición y baja lógica de perfiles (Administrador / Auxiliar) y autenticación.
- Alertas de punto de pedido (visual/listado) para planificar reposición.
- Reportes básicos para la toma de decisiones: movimientos por período, consumos por servicio, stock crítico.
- Auditoría mínima: registro de quién y cuándo realizó cada movimiento.

No contempla

El sistema no contempla facturación ni emisión de facturas o tickets, conexión a servicios de la AFIP, interconexión con sistemas de obras sociales, la trazabilidad regulada de medicamentos ni la gestión de autorizaciones; tampoco incluye la realización de pedidos a laboratorios/proveedores desde la aplicación (solo se registran ingresos ya aprobados). Asimismo, no se permiten modificaciones sobre registros de ingresos o egresos ya confirmados: cualquier corrección deberá realizarse mediante movimientos reversos o ajustes con auditoría.

Elicitación

Para adquirir el conocimiento se realizaron entrevistas en la clínica. En primer lugar, se mantuvieron reuniones con Dirección/Administración, con el fin de relevar el modelo operativo actual y el objetivo que se persigue al finalizar el proyecto. En segundo término, se entrevistó a la Jefatura de Farmacia Central y a Referentes de Servicios (guardia, internación, quirófano) para conocer:

- El proceso operativo de compras y recepción de insumos en Farmacia Central (hasta su ingreso al stock).
- El proceso de almacenamiento y distribución interna desde Farmacia Central hacia los servicios (pedidos internos y egresos de stock).
- Los tipos de usuarios del sistema y sus perfiles de acceso (administrador y auxiliar) según responsabilidades.
- Los criterios internos para stock mínimo, identificación de críticos y manejo básico de vencimientos (sin trazabilidad regulada).

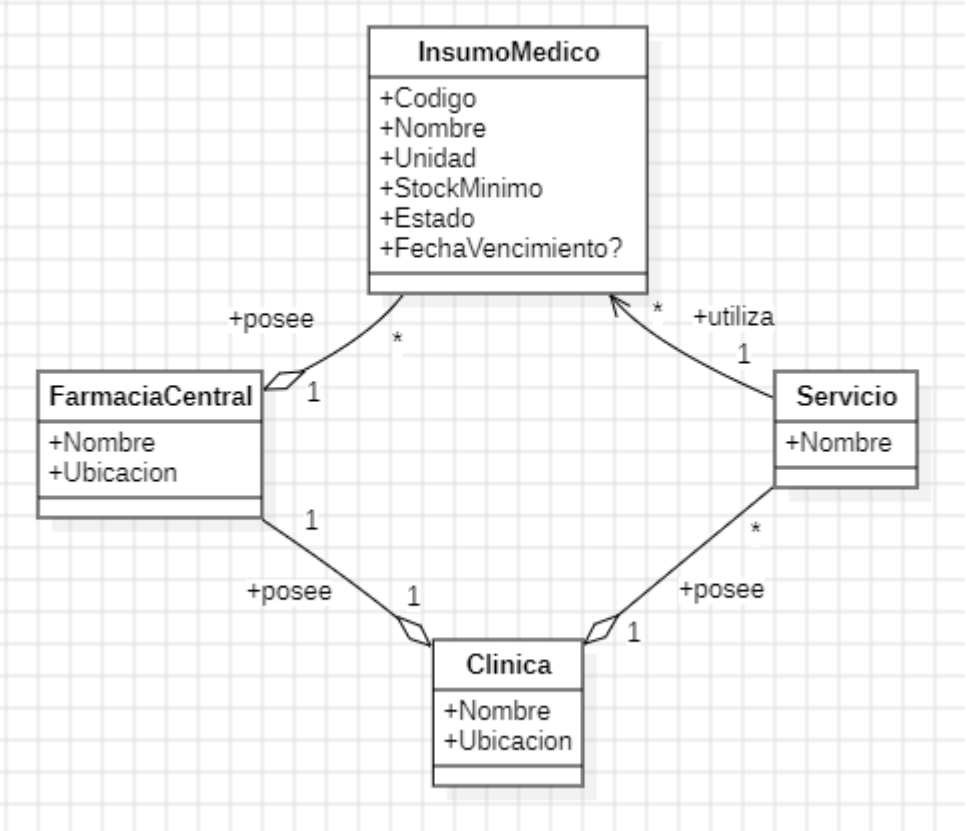
Luego se realizaron visitas a Farmacia Central para observar el circuito completo desde la recepción del insumo hasta la entrega al servicio solicitante, así como relevar:

- El hardware disponible (PCs y conectividad; sin requerir dispositivos especializados).

- El software utilizado actualmente (planillas y/o módulos administrativos no específicos de stock).
- La visión del personal que operará el sistema (expectativas, puntos a mejorar, facilidad de uso).
- El nivel de conocimiento informático del equipo, para adecuar la interfaz y la capacitación.

Como parte del proceso de elicitación se revisaron las políticas internas y procedimientos mínimos aplicables a la gestión de insumos en la clínica (seguridad de acceso, registros y auditoría básica). No se abordaron en esta etapa la trazabilidad regulada, la facturación ni integraciones externas, por encontrarse fuera del alcance definido para la primera iteración del sistema.

Diagrama de dominio



Propuesta de solución funcional

Desarrollar un sistema de gestión de stock para la Farmacia Central de la clínica que registre ingresos (compras aprobadas) y egresos (entregas a servicios: guardia, internación, quirófano y consultorios), mantenga el stock actualizado, administre stock mínimo por insumo y emita alertas cuando se alcance el punto de pedido. El sistema contará con dos perfiles: Administrador (Jefe/a de Farmacia) para configurar insumos, usuarios y reportes; y Auxiliar de Farmacia para operar ingresos/egresos y consultas. Incluirá reportes simples (consumos por período/servicio y críticos) para planificar la reposición y asegurar la continuidad asistencial.

Propuesta de solución técnica

El sistema se implementará en Java con persistencia en MySQL. La interfaz de usuario será una aplicación de escritorio desarrollada con JavaFX (alternativa válida: Swing), y el acceso a datos se realizará mediante JDBC (MySQL Connector/J) utilizando consultas preparadas. Modelo mínimo de tablas: usuarios, insumos, servicios y movimientos. El despliegue será en equipos de la Clínica Horizonte con JRE/JDK instalado; la base de datos residirá en MySQL local o de red interna. Se prevé empaquetado del cliente y configuración de conexión por archivo de propiedades. La solución queda preparada para evolucionar sin cambiar el modelo central.

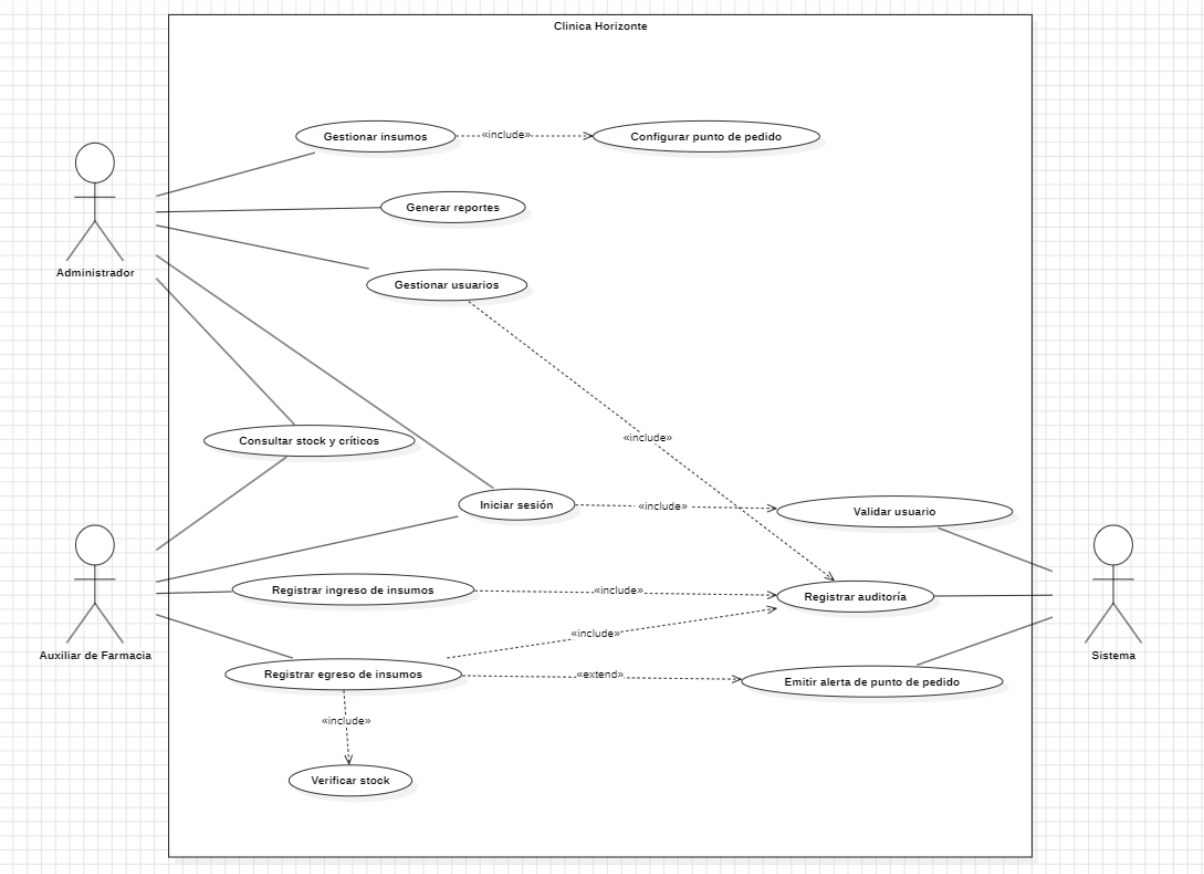
Requerimientos funcionales

Requerimiento	Descripción
RFS01	El sistema debe permitir el ingreso mediante usuario y contraseña y validar el rol (Administrador / Auxiliar de Farmacia).
RFS02	El sistema debe permitir la gestión de usuarios (alta, edición y baja) con asignación de roles.
RFS03	El sistema debe visualizar el listado de usuarios activos con filtros básicos.
RFS04	El sistema debe permitir la gestión de insumos médicos (alta, edición y baja) con código, nombre, unidad, stock mínimo, estado y vencimiento (opcional).
RFS05	El sistema debe permitir buscar insumos por código.
RFS06	El sistema debe permitir buscar insumos por nombre.
RFS07	El sistema debe registrar ingresos de insumos (compras aprobadas) y actualizar el stock.
RFS08	El sistema debe registrar egresos de insumos hacia servicios (guardia, internación, quirófano, consultorios) con descuento automático de stock.
RFS09	El sistema debe consultar el stock disponible y listar críticos (ítems con stock ≤ stock mínimo).
RFS10	El sistema debe emitir alertas de punto de pedido (vista/listado).
RFS11	El sistema debe generar reportes simples: stock crítico (obligatorio) y movimientos por período/servicio (sencillo).
RFS12	El sistema debe registrar auditoría mínima de operaciones (usuario, fecha y acción para altas y movimientos).

Requerimientos no funcionales

Requerimiento	Descripción
RNF01	El sistema debe estar desarrollado en Java.
RNF02	El sistema debe contar con base de datos MySQL.
RNF03	La interfaz de usuario debe ser una aplicación de escritorio con JavaFX (alternativa aceptada: Swing)
RNF04	Compatibilidad: ejecución con JRE/JDK requerido en equipos de la clínica (Windows/Linux).
RNF05	No permitir ingreso de personal no registrado.
RNF06	El sistema debe estar operativo en horario administrativo de Farmacia Central.
RNF07	El acceso a datos debe realizarse vía JDBC con consultas preparadas.

Diagrama de caso de uso



Trazabilidad

Como referencia, se van a seleccionar los siguientes casos de uso para avanzar:

- CU001 Registrar usuario en el sistema
- CU002 Validar usuario en el sistema
- CU003 Gestionar usuarios en el sistema
- CU005 Registrar egreso (consumo) de insumos

Requerimiento	Caso de Uso	Actor	Comentario
RFS02	CU001	Administrador	Realizar alta de usuario
RFS01	CU002	Sistema	Validar ingreso de usuarios
RFS02	CU003	Administrador	Modificar o eliminar usuarios
RFS08	CU005	Auxiliar de Farmacia	Realizar retiro/egreso de insumo

Descripción de casos de uso

CU001	Registrar usuario en el sistema	
Descripción	Este caso de uso le permite al Administrador registrar un nuevo usuario en el sistema.	
Actor	Usuario Administrador.	
Precondición	El usuario a registrar pertenece a la Clínica Horizonte y posee legajo.	
Secuencia	Paso	Acción
	1	El administrador ingresa a configuración.
	2	El sistema solicita usuario y contraseña si no hay sesión activa.
	3	El sistema verifica credenciales del administrador.
	4	El administrador selecciona nuevo usuario.
	5	Ingresa legajo, nombre, apellido, usuario y rol (administrador/auxiliar).
	6	El administrador presiona guardar.
	7	El sistema confirma el alta y habilita el acceso del nuevo usuario.
Postcondición	El nuevo usuario queda registrado y puede autenticarse.	
Excepciones	Paso	Acción
	3	Si los datos son incorrectos, se notifica para reintentar.
	5	Si el legajo o usuario ya existe, se informa el duplicado y se cancela el alta.

CU002	Validar usuario en el sistema	
Descripción	Valida las credenciales de acceso y habilita la sesión según el rol.	
Actor	Sistema	
Precondición	El usuario existe en el sistema.	
Secuencia	Paso	Acción
	1	El usuario ingresa usuario y contraseña en la pantalla de inicio.
	2	El sistema verifica la coincidencia de credenciales.
	3	El sistema inicia sesión y habilita las funciones según el rol.
Postcondición	El usuario accede al sistema de gestión.	

Excepciones	Paso	Acción
	2	Usuario o contraseña incorrectos, se informa y no se inicia sesión.

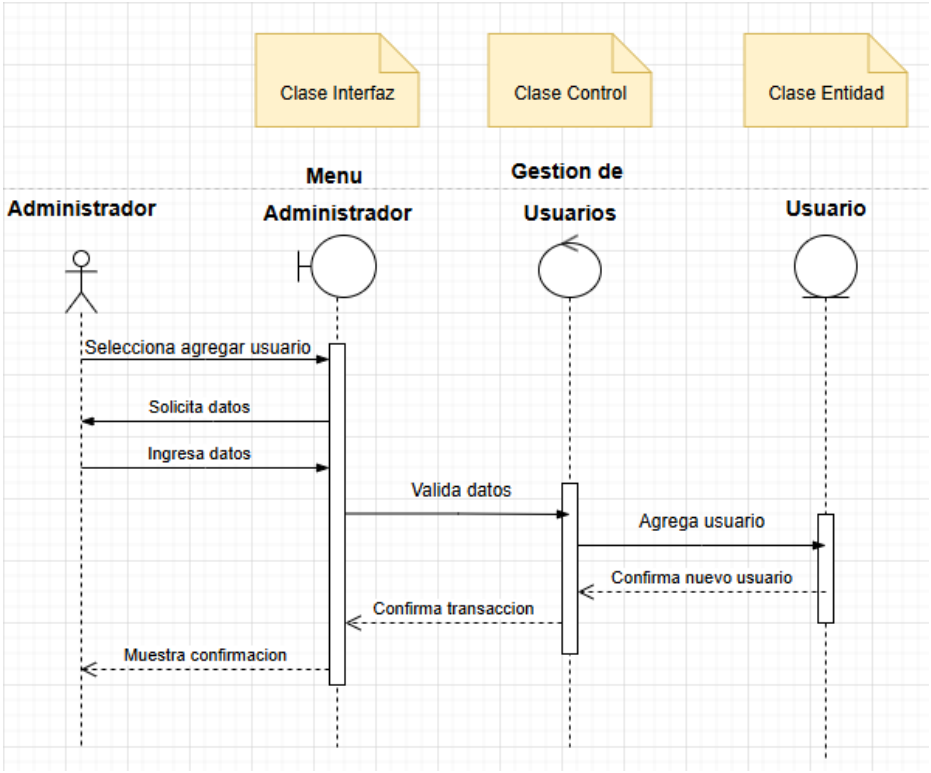
CU003	Gestionar usuarios en el sistema	
Descripción	Permite al administrador modificar o eliminar usuarios existentes.	
Actor	Usuario Administrador.	
Precondición	Administrador autenticado y usuario existente.	
Secuencia	Paso	Acción
	1	Acceder a gestión de usuarios
	2	Seleccionar el usuario objetivo.
	3	Editar o eliminar usuario.
	4	Confirmar la operación.
	5	El sistema registra la modificación o eliminación y confirma la acción.
Postcondición	El usuario queda actualizado o eliminado.	
Excepciones	Paso	Acción
	3	Datos inválidos o usuario inexistente, se informa y se cancela.
	4	El administrador cancela la operación, no se realizan cambios.

CU005	Registrar egreso (consumo) de insumos	
Descripción	Permite al auxiliar de farmacia registrar el egreso de insumos desde farmacia central hacia un servicio.	
Actor	Auxiliar de Farmacia.	
Precondición	Usuario autenticado; El insumo y el servicio existen; hay stock suficiente.	
Secuencia	Paso	Acción
	1	Acceder a registrar egreso.
	2	Buscar y seleccionar el insumo por código o nombre.
	3	Seleccionar el servicio de destino.
	4	Ingresar cantidad y observación (opcional).
	5	El sistema verifica stock disponible.
	6	El sistema descuenta el stock y registra el movimiento.
	7	El sistema confirma el egreso; si el nuevo stock ≤ stock mínimo, muestra alerta de punto de pedido.
Postcondición	Egreso registrado y stock actualizado.	
Excepciones	Paso	Acción
	5	Stock insuficiente, se informa y no se registra el egreso.
	2/3	Insumo o servicio inexistente, se informa y se cancela la operación.

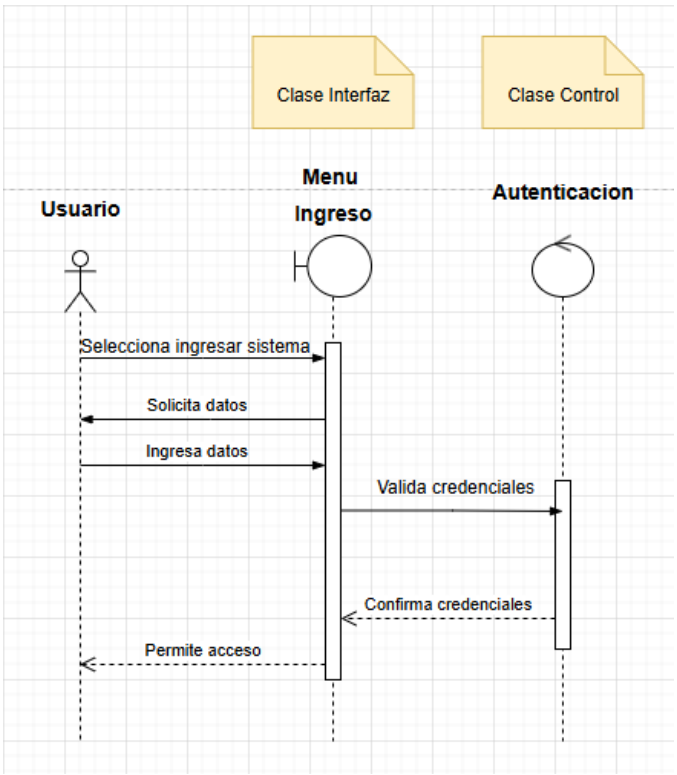
Etapa de análisis

El diagrama de colaboración permite identificar el intercambio de mensajes entre objetos, que representan las operaciones de las clases correspondientes. El sentido del flujo de los mensajes se representa con una flecha entre los objetos. Los mensajes anidados representan una llamada recursiva de una operación o método perteneciente al mismo objeto.

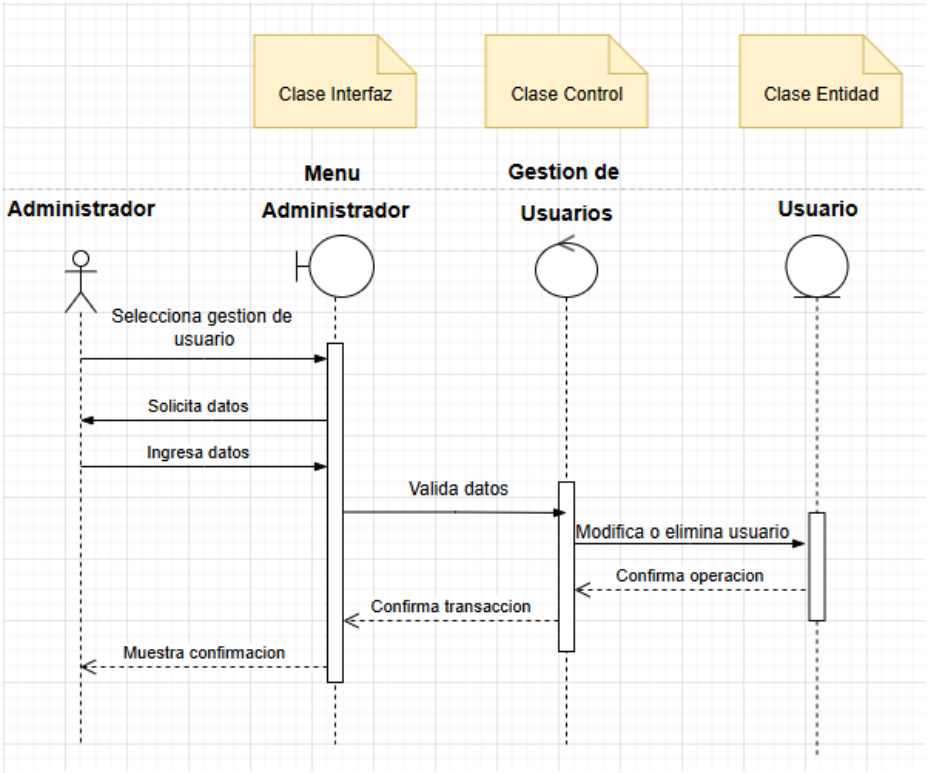
A continuación, se presenta el diagrama de secuencia para el CU001, registrar usuario en el sistema en su curso normal.



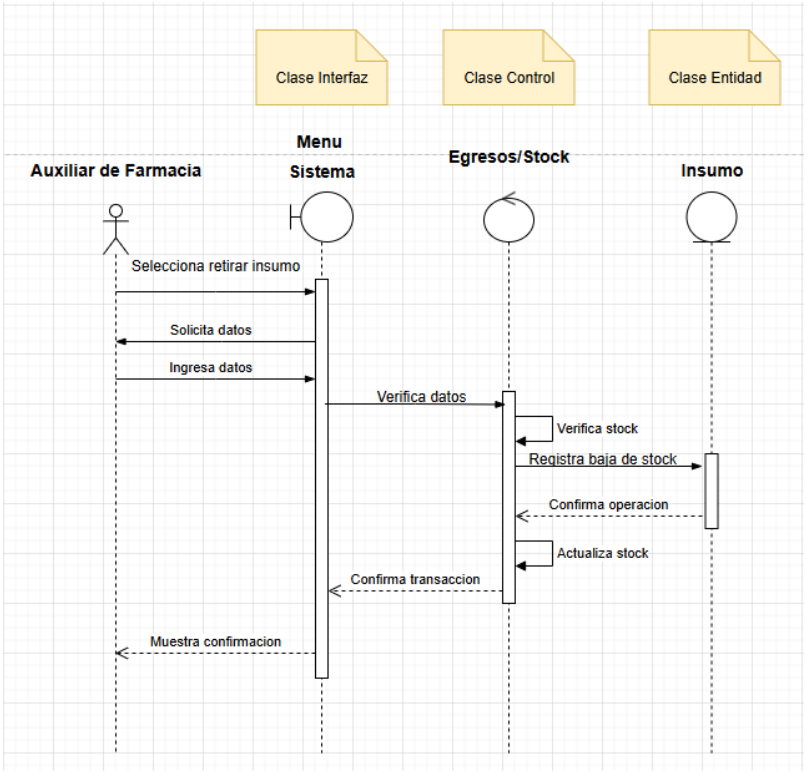
Se presenta el diagrama de secuencia para el CU002, validar usuario en el sistema en su curso normal.



Se presenta el diagrama de secuencia para el CU003, gestionar usuarios en el sistema en su curso normal.

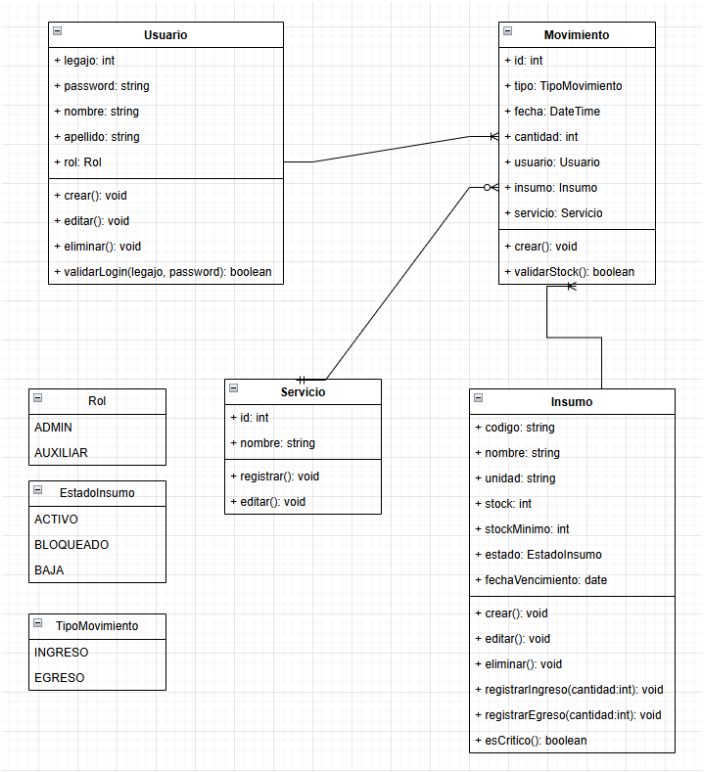


Se presenta el diagrama de secuencia para el CU005, registrar egreso (consumo) de insumos en su curso normal.



Etapa de diseño

Un diagrama de clases de diseño representa las clases de software, sus atributos, operaciones y relaciones. Se utiliza para describir la estructura estática del sistema y cómo interactúan sus partes lógicas. En esta propuesta para la Clínica Horizonte se muestra el modelo de software centrado en entidades, servicios y acceso a datos.



Etapa de implementación

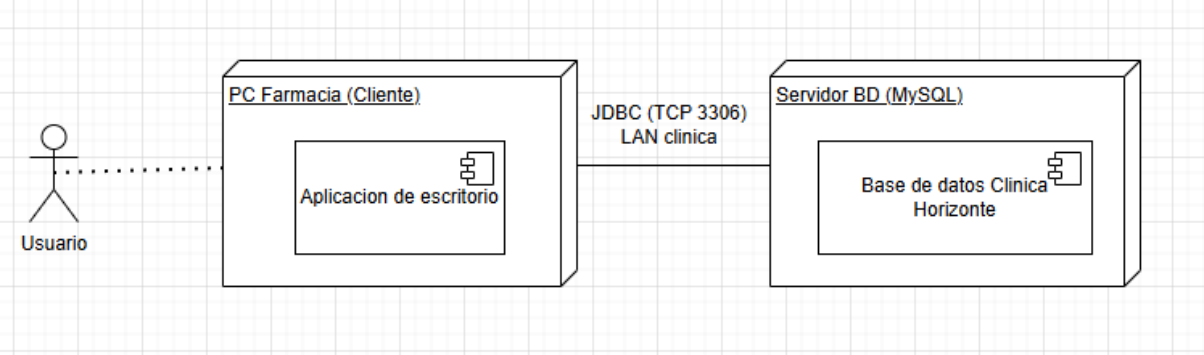
A través de un diagrama de despliegue se representan los componentes del sistema como nodos físicos y sus interacciones. Esto permite entender cómo se va a desplegar la aplicación y cómo se comunican sus partes en la implementación real.

En nuestro caso, la aplicación es de escritorio (JavaFX) que corre en los equipos de Farmacia Central y se conecta por JDBC a una base MySQL alojada en un servidor de la clínica.

Para esta etapa se van a incorporar a los requisitos funcionales (RFS) que se están trabajando, además del siguiente listado de requisitos no funcionales (RNF).

Requerimiento	Descripción
RNF01	El sistema debe estar desarrollado en Java.
RNF02	El sistema debe contar con base de datos MySQL.
RNF03	La interfaz de usuario debe ser una aplicación de escritorio con JavaFX (alternativa aceptada: Swing)

RNF04	Compatibilidad: ejecución con JRE/JDK requerido en equipos de la clínica (Windows/Linux).
RNF05	No permitir ingreso de personal no registrado.
RNF06	El sistema debe estar operativo en horario administrativo de Farmacia Central.
RNF07	El acceso a datos debe realizarse vía JDBC con consultas preparadas.



Etapa de pruebas

Plan de pruebas

Establecer pautas y estrategia para la aceptación del Sistema de Gestión de Stock de Insumos – Clínica Horizonte. Se verificará que el sistema pueda entrar en operación con la totalidad de las funcionalidades priorizadas (login, gestión de usuarios, egresos de insumos, consulta de críticos y reportes simples).

Alcance de las pruebas

Se incluyen pruebas funcionales y no funcionales mínimas sobre los siguientes **módulos**:

Modulo	Contenido
Usuarios	Login, alta/edición/baja lógica, listado.
Stock	ABM de insumos, búsqueda por código/nombre, críticos.
Movimientos	Registro de egresos a servicios y actualización de stock.
Reportes	Movimientos por periodo/servicio y stock crítico.

Objetivo de las pruebas

Validar: (a) visualización de datos ingresados/modificados, (b) operación de funciones según necesidades planteadas y (c) consistencia de transacciones (descuento/ajuste de stock y registro de movimientos).

Orden de ejecución de los módulos

1. Usuarios → 2) Stock → 3) Movimientos → 4) Reportes.

Responsabilidades

El equipo de testing ejecuta y documenta las pruebas en conjunto con usuarios clave (Administrador de Farmacia y Auxiliar de Farmacia). El equipo de desarrollo atiende incidencias y aplica correcciones.

Casos de pruebas incluidos

Se toman como referencia los CU definidos: CU001, CU002, CU003, CU005.

CU	ID	Nivel	Técnica	Alcance
CU005	CPF05	Componente	Caja negra – partición/equivalencia y frontera	Método registrarEgreso(cantidad) del Insumo.
CU005	CPC05	Componente	Caja blanca – cobertura	Ramas del descuento de stock y disparo de alerta por punto de pedido.
CU005	CPS05	Sistema	Prueba funcional	Flujo completo: seleccionar insumo/servicio, cantidad, confirmar y verificar stock/movimiento.
CU002	CPS02	Sistema	Funcional	Credenciales validas/invalidas y control de rol.
CU001	CPS01	Sistema	Funcional	Alta correcta, legajo duplicado, campos obligatorios.
CU003	CPS03	Sistema	Funcional	Edición y baja lógica con reflejo en listado.

Entorno y configuración de las pruebas

- Cliente: PC Farmacia con JavaFX (JDK/JRE 17+), SO Windows 10/11 o Ubuntu 22.04.
- Servidor BD: MySQL 8.x en servidor de la clínica (LAN).
- Conexión: JDBC (MySQL Connector/J) puerto TCP 3306.
- Base de pruebas: clinica_horizonte_test.
- Datos semilla:
    - Usuarios: admin (legajo 1000, rol ADMIN), aux1 (legajo 2000, rol AUXILIAR).
    - Servicios: Guardia, Internación, Quirófano, Consultorios.
    - Insumos: COD-A Guantes (stock=50, mín=10), COD-B Gasas (stock=12, mín=10).

Criterios de inicio

- Aprobación del plan de pruebas y de los casos.
- Entrega de la build de prueba y scripts de BD.
- Ambiente cliente/servidor configurado y verificado.

Criterios de aprobación / rechazo

- Graves: datos críticos mal grabados, caída de app, no descuento de stock, login inoperante.
- Medios: errores de presentación o en funciones secundarias.

- Leves: mensajes/estilos, usabilidad menor.

Estrategia de pruebas

Escenarios

1. Pruebas de instalación: arranque de la app, lectura de configuración y conexión JDBC correcta contra MySQL.
2. Pruebas de GUI (JavaFX): navegación, foco, validaciones (campos obligatorios y valores límite), mensajes, usabilidad básica.
3. Pruebas funcionales: ejecución de CU end-to-end (login, ABM usuarios, egreso, críticos, reportes).

Orden de ejecución

1. Configurar BD y cadena de conexión JDBC.
2. Crear usuarios de prueba (Administración).
3. Registrar egresos que produzcan: (a) operación válida, (b) stock insuficiente, (c) punto de pedido (stock ≤ mínimo).
4. Verificar listado de críticos y reporte por período/servicio.

Equipo de pruebas y responsabilidades

Rol	Responsabilidad
Arquitecto/Dev líder	Define condiciones de termino y autoriza cierre.
QA/Test	Diseña/ejecuta casos, documenta evidencias e incidencias.
Analista Funcional	Aclara reglas, valida resultados con usuarios.
Usuario Clave (Admin/Aux)	Pruebas de aceptación y validación operativa.

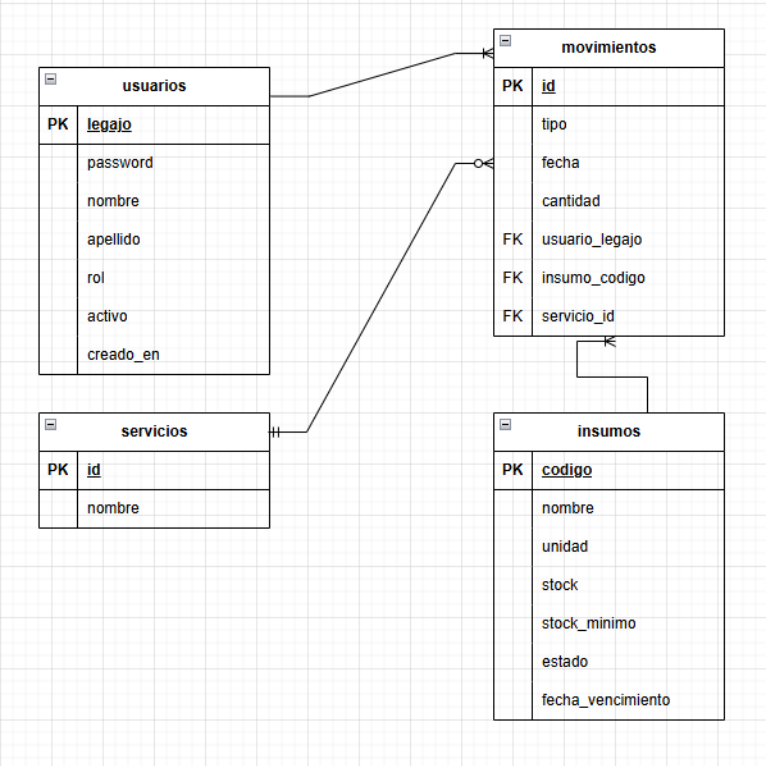
Definición de base de datos para el sistema

El sistema usará una base de datos relacional MySQL para guardar de forma segura la información necesaria. Allí se registrarán usuarios (para el acceso), insumos (código, nombre, unidad, stock y stock mínimo), servicios internos (Guardia, Internación, etc.) y movimientos de ingreso/egreso que actualizan el stock.

Con estos datos se podrán hacer consultas e informes básicos: stock disponible, lista de críticos (stock ≤ mínimo) y movimientos por período/servicio. MySQL se elige por su buen rendimiento, facilidad de uso y porque asegura la consistencia entre los datos.

Diagrama Entidad – Relación

El DER representa la estructura de la base de datos y cómo se relacionan sus tablas.
La base de datos se llamará clinica_horizonte y, siguiendo el enfoque simple del proyecto, se definen cuatro tablas principales que cubren el alcance: usuarios, insumos, servicios y movimientos. Con ellas identificamos los atributos clave de catálogo (usuarios, insumos, servicios) y registramos la trazabilidad de los egresos/ingresos de stock mediante movimientos.



Creación de tablas

A continuación, se presentará la creación de la base de datos y cada una de las tablas que la componen.

Creación de la base de datos llamada clinica_horizonte

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0234 seconds.)

```
CREATE DATABASE IF NOT EXISTS clinica_horizonte DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_unicode_ci;
```

Creación de tabla usuarios

MySQL returned an empty result set (i.e. zero rows). (Query took 0.2431 seconds.)

```
CREATE TABLE usuarios ( legajo INT NOT NULL, password VARCHAR(255) NOT NULL, nombre VARCHAR(60) NOT NULL, apellido VARCHAR(60) NOT NULL, rol ENUM('ADMIN','AUXILIAR') NOT NULL, activo TINYINT(1) NOT NULL DEFAULT 1, creado_en DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (legajo) ) ENGINE=InnoDB;
```

Creación de tabla servicios

MySQL returned an empty result set (i.e. zero rows). (Query took 0.5889 seconds.)

```
CREATE TABLE servicios ( id INT NOT NULL AUTO_INCREMENT, nombre VARCHAR(60) NOT NULL, PRIMARY KEY (id), UNIQUE KEY uk_servicio_nombre (nombre) ) ENGINE=InnoDB;
```

Creación de tabla insumos

MySQL returned an empty result set (i.e. zero rows). (Query took 0.2426 seconds.)

```
CREATE TABLE insumos ( codigo VARCHAR(30) NOT NULL, nombre VARCHAR(120) NOT NULL, unidad VARCHAR(20) NOT NULL, stock INT UNSIGNED NOT NULL DEFAULT 0, stock_minimo INT UNSIGNED NOT NULL DEFAULT 0, estado ENUM('ACTIVO','BLOQUEADO','BAJA') NOT NULL DEFAULT 'ACTIVO', fecha_vencimiento DATE NULL, PRIMARY KEY (codigo), CHECK (stock >= 0), CHECK (stock_minimo >= 0) ) ENGINE=InnoDB;
```

Creación de tabla movimientos

MySQL returned an empty result set (i.e. zero rows). (Query took 0.7633 seconds.)

```
CREATE TABLE movimientos ( id BIGINT NOT NULL AUTO_INCREMENT, tipo ENUM('INGRESO','EGRESO') NOT NULL, fecha DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP, cantidad INT UNSIGNED NOT NULL, usuario_legajo INT NOT NULL, insumo_codigo VARCHAR(30) NOT NULL, servicio_id INT NULL, PRIMARY KEY (id), KEY ix_mov_fecha (fecha), KEY ix_mov_insumo (insumo_codigo), KEY ix_mov_serv (servicio_id), CONSTRAINT fk_mov_usuario FOREIGN KEY (usuario_legajo) REFERENCES usuarios(legajo) ON UPDATE CASCADE ON DELETE RESTRICT, CONSTRAINT fk_mov_insumo FOREIGN KEY (insumo_codigo) REFERENCES insumos(codigo) ON UPDATE CASCADE ON DELETE RESTRICT, CONSTRAINT fk_mov_servicio FOREIGN KEY (servicio_id) REFERENCES servicios(id) ON UPDATE CASCADE ON DELETE RESTRICT, CHECK (cantidad > 0), CHECK ( (tipo='INGRESO' AND servicio_id IS NULL) OR (ti[...]
```

Inserción, consulta y borrado de registros

Se van a insertar datos de prueba en algunas tablas para verificar que el modelo de datos y las funcionalidades de la base de datos funcionen correctamente antes de implementar la aplicación en un entorno de producción. Luego, se mostrará con una consulta cómo obtener los resultados y finalmente, se procederá a limpiar las tablas.

Inserción de datos de pruebas

```sql
INSERT INTO usuarios (legajo, password, nombre, apellido, rol, activo)
VALUES
  (1000, 'admin123', 'Ana',  'García', 'ADMIN',   1),
  (2000, 'aux123',   'Luis', 'Pérez',  'AUXILIAR',1);
```

```sql
INSERT INTO servicios (nombre)
VALUES
  ('Guardia'),
  ('Internación'),
  ('Quirófano'),
  ('Consultorios');
```

```sql
INSERT INTO insumos (codigo, nombre, unidad, stock, stock_minimo, estado, fecha_vencimiento)
VALUES
  ('GUA-01', 'Guantes de látex',   'caja',    50, 10, 'ACTIVO',    NULL),
  ('GAS-01', 'Gasas estériles',    'pack',    12, 10, 'ACTIVO',    NULL),
  ('ALC-70', 'Alcohol 70%',        'botella',  5,  5, 'ACTIVO',    '2026-06-30'),
  ('JER-05', 'Jeringas 5 ml',      'unidad', 100, 40, 'ACTIVO',    NULL),
  ('BAR-01', 'Barbijo quirúrgico', 'caja',    25, 20, 'ACTIVO',    NULL);
```

```sql
INSERT INTO movimientos (tipo, cantidad, usuario_legajo, insumo_codigo, servicio_id)
VALUES ('INGRESO', 10, 1000, 'GAS-01', NULL);

INSERT INTO movimientos (tipo, cantidad, usuario_legajo, insumo_codigo, servicio_id)
VALUES ('EGRESO', 5, 2000, 'GUA-01',
        (SELECT id FROM servicios WHERE nombre='Guardia'));

INSERT INTO movimientos (tipo, cantidad, usuario_legajo, insumo_codigo, servicio_id)
VALUES ('EGRESO', 8, 2000, 'BAR-01',
        (SELECT id FROM servicios WHERE nombre='Consultorios'));
```

Consulta de los datos

```sql
SELECT
  m.id, m.fecha, m.tipo, m.cantidad,
  i.codigo  AS insumo_codigo,
  i.nombre  AS insumo_nombre,
  i.unidad,
  s.nombre  AS servicio,
  u.legajo,
  CONCAT(u.nombre, ' ', u.apellido) AS usuario,
  u.rol
FROM movimientos m
JOIN insumos   i ON i.codigo = m.insumo_codigo
LEFT JOIN servicios s ON s.id = m.servicio_id
JOIN usuarios  u ON u.legajo = m.usuario_legajo
ORDER BY m.fecha DESC, m.id DESC;
```

id	fecha	tipo	cantidad	insumo_codigo	insumo_nombre	unidad	servicio	legajo	usuario	rol
3	2025-10-06 00:45:46	EGRESO	8	BAR-01	Barbijo quirúrgico	caja	Consultorios	2000	Luis Pérez	AUXILIAR
2	2025-10-06 00:45:46	EGRESO	5	GUA-01	Guantes de látex	caja	Guardia	2000	Luis Pérez	AUXILIAR
1	2025-10-06 00:45:46	INGRESO	10	GAS-01	Gasas estériles	pack	*NULL*	1000	Ana García	ADMIN

Borrado de registros

```sql
DELETE FROM movimientos;
DELETE FROM insumos;
DELETE FROM servicios;
DELETE FROM usuarios;
```

```sql
SELECT
  (SELECT COUNT(*) FROM usuarios)    AS usuarios,
  (SELECT COUNT(*) FROM servicios)   AS servicios,
  (SELECT COUNT(*) FROM insumos)     AS insumos,
  (SELECT COUNT(*) FROM movimientos) AS movimientos;
```

usuarios	servicios	insumos	movimientos
0	0	0	0

Desarrollo del sistema utilizando Java

Explicación del desarrollo en Java

De acuerdo con los requerimientos no funcionales, el sistema se desarrolla en Java. Se presenta un sistema funcional completo que implementa persistencia con JDBC/MySQL, manteniendo un diseño modular, legible y escalable.El sistema implementa los casos de uso CU002 (Login), CU001/CU003 (Gestión de usuarios) y CU005 (Registro de egresos/ingresos) utilizando repositorios JDBC que

persisten datos en una base de datos MySQL. La arquitectura en capas y el uso de interfaces de repositorios mantienen la lógica de negocio desacoplada de la persistencia, facilitando el mantenimiento y futuras mejoras.

Arquitectura y organización

El proyecto sigue una arquitectura en capas bien definida, implementando varios patrones de diseño.Patrón de Capas:

- Presentación (app/) - Handlers y UI
- Lógica de Negocio (usecase/) - Servicios de negocio
- Repositorios (repo/) - Persistencia con implementaciones JDBC/MySQL
- Entidades (domain/) - Modelo de dominio

Patrones de Diseño:

- Repository Pattern - Abstracción de persistencia con implementaciones JDBC
- Service Layer Pattern - Encapsulación de lógica de negocio
- Dependency Injection - Inyección por constructor
- Strategy Pattern - Interfaces intercambiables de repositorios
- Factory Pattern - Inicialización centralizada
- Transaction Manager Pattern - Gestión de transacciones JDBC
- DAO Pattern - Acceso a datos mediante repositorios JDBC
- Handler Pattern - Separación por dominio funcional

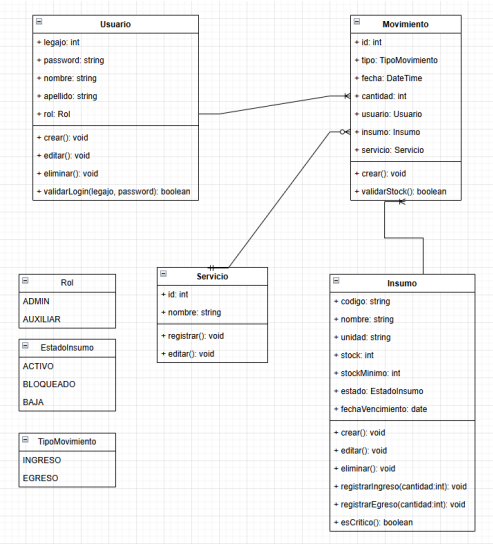
La persistencia se implementa con JDBC/MySQL, utilizando transacciones para garantizar la integridad de los datos.

Estructura:

```
clinica-horizonte/
├── app/              # Capa de presentación
│   ├── handlers/     # Manejadores de operaciones (separados por dominio)
│   │   ├── AuthHandler.java        # Autenticación y login
│   │   ├── UsuarioHandler.java     # Gestión de usuarios (CRUD)
│   │   ├── StockHandler.java       # Operaciones de stock e insumos
│   │   └── ReporteHandler.java     # Generación de reportes
│   ├── ui/           # Interfaz de usuario
│   │   ├── ConsoleUI.java          # Utilidades de entrada/salida
│   │   └── MenuPrincipal.java      # Constantes del menú
│   └── MainDemo.java               # Punto de entrada y orquestación
├── bin/              # Archivos compilados (.class)
├── domain/           # Entidades del dominio
│   ├── enums/        # Enumeraciones (Rol, EstadoInsumo, TipoMovimiento)
│   ├── Usuario.java
│   ├── Persona.java
│   ├── Insumo.java
│   ├── Movimiento.java
│   └── Servicio.java
├── repo/             # Capa de persistencia
│   ├── jdbc/         # Implementaciones JDBC (MySQL)
│   │   ├── DatabaseConnection.java
│   │   ├── TransactionManager.java
│   │   ├── UsuarioJDBC.java
│   │   ├── InsumoJDBC.java
│   │   ├── MovimientoJDBC.java
│   │   └── ServicioJDBC.java
│   ├── memory/       # Implementaciones en memoria (legacy)
│   │   ├── UsuarioInMemory.java
│   │   ├── InsumoInMemory.java
│   │   ├── MovimientoInMemory.java
│   │   └── ServicioInMemory.java
│   ├── UsuarioRepository.java
│   ├── InsumoRepository.java
│   ├── MovimientoRepository.java
│   └── ServicioRepository.java
├── usecase/          # Lógica de negocio
│   ├── AutenticacionService.java
│   ├── GestionUsuariosService.java
│   ├── StockService.java
│   └── ReportesService.java
├── exceptions/       # Excepciones personalizadas
│   ├── CredencialesInvalidasException.java
│   ├── StockInsuficienteException.java
│   ├── EntidadNoEncontradaException.java
│   └── DatabaseException.java
├── lib/              # Librerías externas
│   └── mysql-connector-j-*.jar  # Driver de MySQL
├── clinica_horizonte.sql  # Script de creación de BD
├── compile.sh        # Script de compilación
└── run.sh            # Script de ejecución
```

Modelo de dominio

El diagrama de clases muestra las entidades principales del sistema



Entidades principales

Usuario

La clase Usuario implementa el control de acceso mediante roles (ADMIN/AUXILIAR), permitiendo gestionar permisos específicos para cada tipo de usuario.

```java
public class Usuario extends Persona {
  private int legajo;
  private String password; // por ahora texto plano
  private Rol rol;
  private boolean activo = true;

  public Usuario(int legajo, String password, String nombre, String
  apellido, Rol rol) {
    super(nombre, apellido);
    this.legajo = legajo;
    this.password = password;
    this.rol = rol;
  }
```

```java
  public int getLegajo() {
    return legajo;
  }

  public String getPassword() {
    return password;
  }

  public Rol getRol() {
    return rol;
  }

  public boolean isActivo() {
    return activo;
  }

  public void desactivar() {
    this.activo = false;
  }

  public void activar() {
    this.activo = true;
  }

  public void setPassword(String password) {
    if (password == null || password.length() < 6) {
      throw new IllegalArgumentException("La contraseña debe tener a
      menos 6 caracteres");
    }
    this.password = password;
  }

  public void setRol(Rol rol) {
    if (rol == null) {
      throw new IllegalArgumentException("El rol no puede ser nulo")
    }
    this.rol = rol;
  }

  @Override
  public String toString() {
    return String.format("Usuario[legajo=%d, nombre=%s %s, rol=%s,
    activo=%s]",
        legajo, getNombre(), getApellido(), rol, activo);
  }
```

Insumos

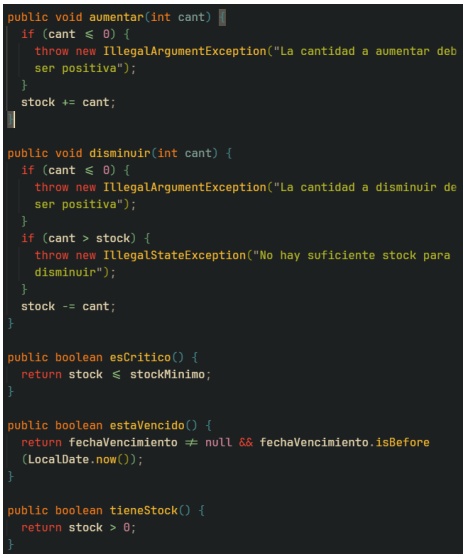
La clase Insumo es central en el sistema, manejando:

- Control de stock
- Validaciones de stock mínimo
- Estados del insumo (ACTIVO, BLOQUEADO, BAJA)
- Fechas de vencimiento

```java
public class Insumo {
  private String codigo, nombre, unidad;
  private int stock, stockMinimo;
  private EstadoInsumo estado = EstadoInsumo.ACTIVO;
  private LocalDate fechaVencimiento; // opcional

  public Insumo(String codigo, String nombre, String unidad, int sto
  int stockMinimo, EstadoInsumo estado,
    LocalDate fechaVencimiento) {
    if (codigo == null || codigo.trim().isEmpty()) {
      throw new IllegalArgumentException("El código no puede estar
      vacio");
    }
    if (nombre == null || nombre.trim().isEmpty()) {
      throw new IllegalArgumentException("El nombre no puede estar
      vacio");
    }
    if (unidad == null || unidad.trim().isEmpty()) {
      throw new IllegalArgumentException("La unidad no puede estar
      vacia");
    }
    if (stock < 0) {
      throw new IllegalArgumentException("El stock no puede ser
      negativo");
    }
    if (stockMinimo < 0) {
      throw new IllegalArgumentException("El stock mínimo no puede s
      negativo");
    }
    if (estado == null) {
      throw new IllegalArgumentException("El estado no puede ser nul
    }

    this.codigo = codigo.trim();
    this.nombre = nombre.trim();
    this.unidad = unidad.trim();
    this.stock = stock;
    this.stockMinimo = stockMinimo;
    this.estado = estado;
    this.fechaVencimiento = fechaVencimiento;
  }
```



```java
public void aumentar(int cant) {
  if (cant <= 0) {
    throw new IllegalArgumentException("La cantidad a aumentar deb
    ser positiva");
  }
  stock += cant;
}

public void disminuir(int cant) {
  if (cant <= 0) {
    throw new IllegalArgumentException("La cantidad a disminuir de
    ser positiva");
  }
  if (cant > stock) {
    throw new IllegalStateException("No hay suficiente stock para
    disminuir");
  }
  stock -= cant;
}

public boolean esCritico() {
  return stock <= stockMinimo;
}

public boolean estaVencido() {
  return fechaVencimiento != null && fechaVencimiento.isBefore
  (LocalDate.now());
}

public boolean tieneStock() {
  return stock > 0;
}
```

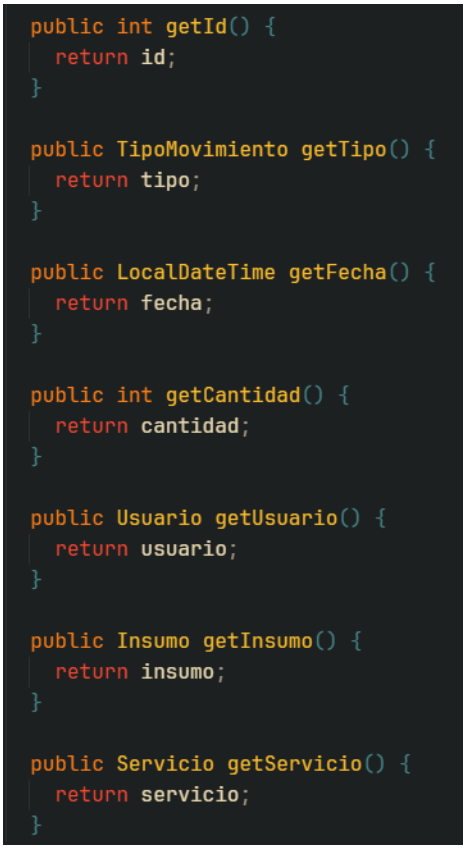
Movimientos

Los Movimientos registran todas las operaciones de stock, vinculando:

- Quién realizó el movimiento (Usuario)
- Qué insumo se movió
- A qué servicio se destinó (en caso de EGRESO)

```java
public class Movimiento {
  private int id;
  private TipoMovimiento tipo;
  private LocalDateTime fecha;
  private int cantidad;
  private Usuario usuario;
  private Insumo insumo;
  private Servicio servicio; // null si INGRESO

  public Movimiento(int id, TipoMovimiento tipo, LocalDateTime fecha
  int cantidad,
      Usuario usuario, Insumo insumo, Servicio servicio) {
    this.id = id;
    this.tipo = tipo;
    this.fecha = fecha;
    this.cantidad = cantidad;
    this.usuario = usuario;
    this.insumo = insumo;
    this.servicio = servicio;
  }
```



```java
  public int getId() {
    return id;
  }

  public TipoMovimiento getTipo() {
    return tipo;
  }

  public LocalDateTime getFecha() {
    return fecha;
  }

  public int getCantidad() {
    return cantidad;
  }

  public Usuario getUsuario() {
    return usuario;
  }

  public Insumo getInsumo() {
    return insumo;
  }

  public Servicio getServicio() {
    return servicio;
  }
```

Servicios

La clase Servicio representa las diferentes áreas de la clínica donde se utilizan los insumos, como Guardia, Internación, Quirófano, etc. Su simplicidad refleja una buena práctica de diseño: cada clase debe tener una única responsabilidad.

```java
public class Servicio {
  private int id;
  private String nombre;

  public Servicio(int id, String nombre) {
    this.id = id;
    this.nombre = nombre;
  }

  public int getId() {
    return id;
  }

  public String getNombre() {
    return nombre;
  }
}
```

Excepciones y control de flujo

Una excepción es una situación anormal que ocurre durante la ejecución de un programa y que interrumpe su flujo normal. Por eso, se manejan excepciones específicas para evitar errores y mejorar la experiencia del usuario.

```java
public class CredencialesInvalidasException extends RuntimeException {
  public CredencialesInvalidasException() {
    super("Credenciales inválidas");
  }

  public CredencialesInvalidasException(String message) {
    super(message);
  }

  public CredencialesInvalidasException(String message, Throwable cause) {
    super(message, cause);
  }
}
```

```java
public class StockInsuficienteException extends RuntimeException {
  public StockInsuficienteException() {
    super("Stock insuficiente");
  }

  public StockInsuficienteException(String message) {
    super(message);
  }

  public StockInsuficienteException(String message, Throwable cause) {
    super(message, cause);
  }
}
```

```java
public class EntidadNoEncontradaException extends RuntimeException {
  public EntidadNoEncontradaException(String msg) {
    super(msg);
  }
}
```

```java
if (ins.getStock() < cant) {
  throw new StockInsuficienteException("Stock insuficiente. Disponible: " + ins.getStock
}
```

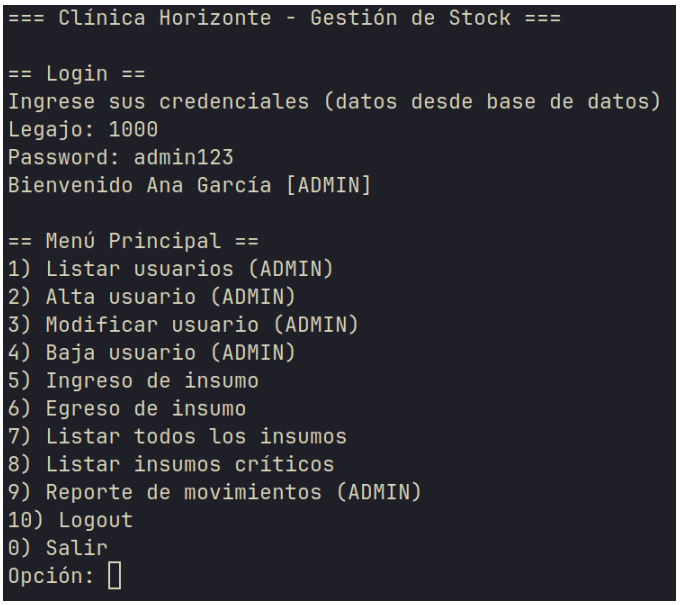
```java
try {
   // Actualizar stock
   insumos.update(ins);

   // Registrar movimiento
   Movimiento mov = new Movimiento(0, TipoMovimiento.INGRESO, LocalDateTime.now(), cant,
   ins, null);
   movimientos.save(mov);
} catch (Exception e) {
   // En el futuro, con JDBC, aquí manejaríamos el rollback de la transacción
   throw new RuntimeException("Error al registrar el ingreso", e);
}
```

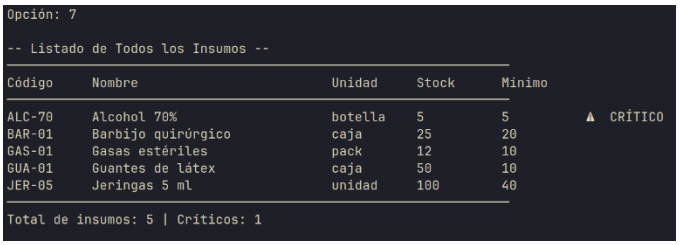
Interfaz de usuario

El menú principal permite gestionar distintas operaciones del sistema desde una interfaz de consola simple y guiada. Tras el login, el sistema identifica el rol del usuario (ADMIN o AUXILIAR) y habilita solo las opciones correspondientes. La navegación se realiza mediante números y se valida cada entrada para evitar errores comunes.

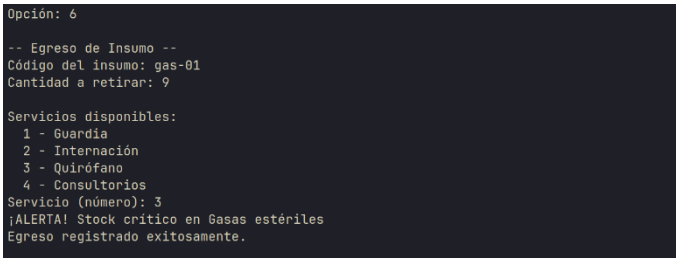
Login y menú principal



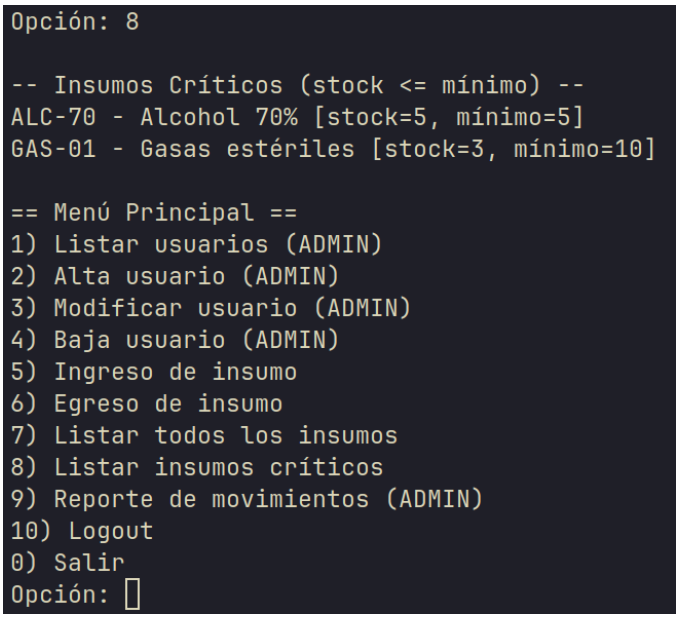
Listado de insumos



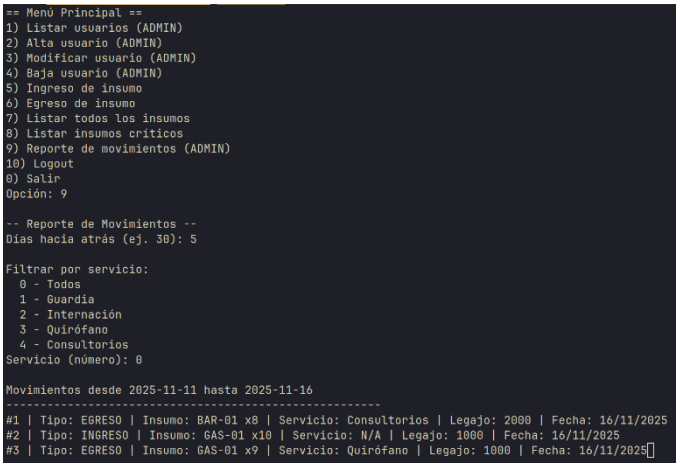
Egreso de insumo



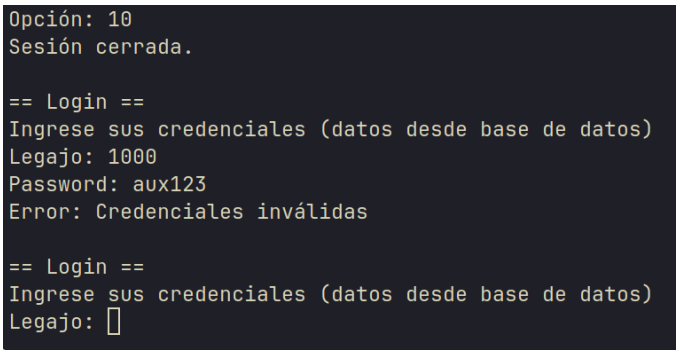
Listado de insumos críticos



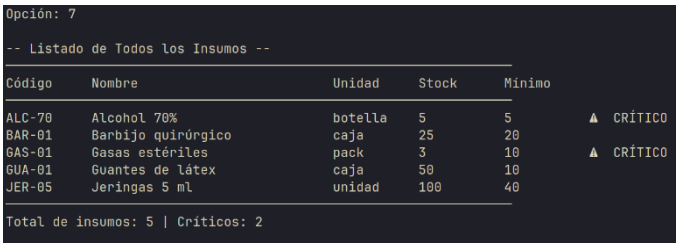
Reporte de movimientos



Login invalido



Listado de insumos actualizado



Para más detalles sobre la implementación de POO (encapsulamiento, herencia, polimorfismo y abstracción), el manejo de excepciones (try/catch específicos), las estructuras de control (condicionales y bucles), las validaciones de entrada y la trazabilidad de movimientos, el código completo del proyecto se encuentra en GitHub.

https://github.com/joaquinclg/clinica-horizonte