Trabajo Final Integrador (TFI)

Informe Técnico - Trabajo Práctico Integrador

Integrantes:

- Nidia Samaniego: Parte del código, parte del informe
- Iván Sierra: Parte del código, pruebas, grabación, parte del informe
- Cristian Siles: Parte del código, pruebas, parte del informe
- Facundo Archiria: Diagrama UML, parte del informe

  Link video: https://youtu.be/EZ--aU8dmZU

1. Introducción

Este trabajo práctico integrador consiste en el desarrollo de un sistema de gestión hospitalaria orientado a la gestión de pacientes y sus historias clínicas. La aplicación está desarrollada en Java, utilizando JDBC para la persistencia en base de datos MySQL y una arquitectura en capas (DAO, Service, Menu/Handler).

2. Arquitectura y Diseño

2.1. Estructura de Paquetes

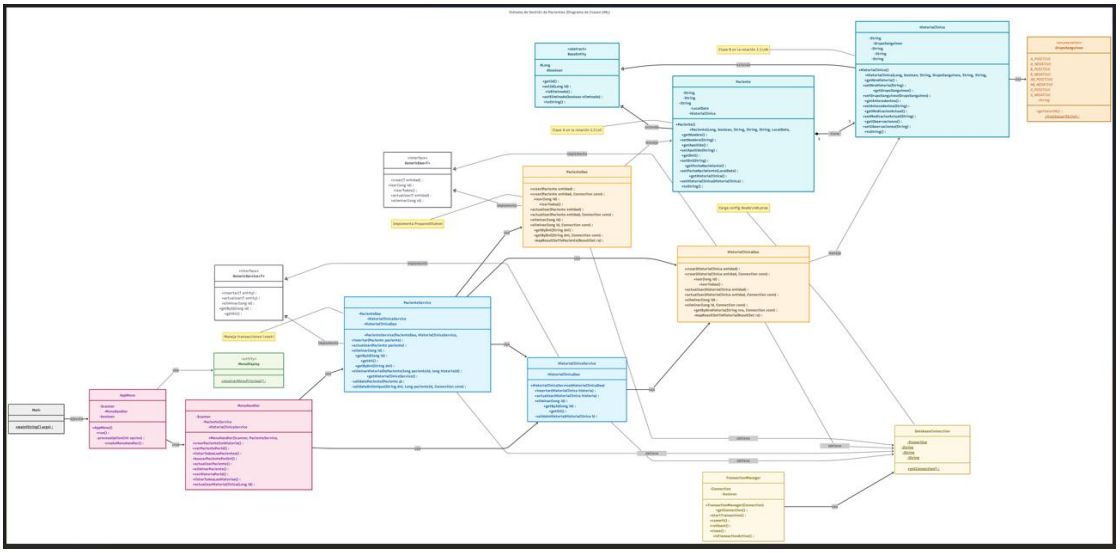
- **entities:** Contiene las entidades del dominio (Paciente, HistoriaClinica, GrupoSanguineo) y la clase base BaseEntity.
- **dao:** Implementación del patrón DAO para acceso a datos (PacienteDao, HistoriaClinicaDao), con una interfaz genérica GenericDao.
- **service:** Servicios que contienen la lógica de negocio y las

transacciones (PacienteService, HistoriaClinicaService).

- **menu:** Manejo de la interfaz de usuario por consola (AppMenu, MenuHandler, MenuDisplay).

- **config:** Configuración de la conexión a base de datos y manejo de transacciones (DatabaseConnection, TransactionManager).

2.2. Diagrama UML



El diagrama representa las relaciones entre entidades, DAOs, servicios y menú. Se destaca la asociación entre Paciente y HistoriaClinica, y la estructura jerárquica con herencia de BaseEntity.

3. Implementación

3.1. Persistencia y DAO

- Se implementó una interfaz genérica GenericDao<T> con métodos CRUD básicos.

- La clase PacienteDao y HistoriaClinicaDao implementan esta interfaz, utilizando JDBC con PreparedStatement para prevenir inyección SQL y manejar parámetros.
- Se implementó baja lógica mediante un campo booleano eliminado para evitar borrados físicos.
- Las consultas de lectura utilizan JOIN para obtener la historia clínica junto con el paciente.

3.2. Entidades

- Paciente y HistoriaClinica extienden de BaseEntity que contiene id y eliminado.
- GrupoSanguineo es un enum con valores mapeados a strings para la base de datos, con método fromValue para conversión segura.

3.3. Servicios

- Los servicios (PacienteService, HistoriaClinicaService) encapsulan la lógica de negocio, validaciones y manejo de transacciones.
- Se usan transacciones para mantener la integridad entre paciente e historia clínica en operaciones de inserción y actualización.
- Se valida unicidad de DNI y número de historia clínica para evitar duplicados.

3.4. Interfaz de Usuario

- La aplicación se maneja por consola, con un menú principal

(AppMenu) y un MenuHandler que recibe las opciones y llama a los servicios correspondientes.

- Se implementaron las opciones: crear paciente con historia, ver paciente/historia, listar, buscar, actualizar y eliminar (baja lógica).
- En la actualización se permite dejar campos en blanco para mantener valores actuales.
- Se manejan excepciones para entradas inválidas y errores en base de datos.

4. Pruebas

4.1. Pruebas Manuales

- Se verificó la creación exitosa de pacientes con historias clínicas, incluyendo validaciones de campos obligatorios y unicidad.
- Se probaron búsquedas por ID y DNI, listados completos y manejo correcto de baja lógica.
- Se validó la actualización de datos tanto para paciente como para historia clínica, comprobando que los datos antiguos se mantienen si se deja vacío.
- Se validaron mensajes de error para entradas inválidas (por ejemplo, formato de fecha, IDs no numéricos).

4.2. Resultado esperado

- Los datos se almacenan y recuperan correctamente desde la base de datos.

- No se permiten duplicados de DNI ni número de historia clínica.
- El sistema responde adecuadamente a errores y confirma operaciones exitosas.

5. Capturas de Código

5.1. Creación de Paciente con Historia Clínica (MenuHandler.java)

```
public void crearPacienteConHistoria() {
    System.out.println("\n--- Creando Nuevo Paciente y su Historia Clínica ---");
    try {
        // Pedimos datos Paciente
        System.out.print("Nombre del Paciente: ");
        String nombre = scanner.nextLine().trim();
        System.out.print("Apellido del Paciente: ");
        String apellido = scanner.nextLine().trim();
        System.out.print("DNI del Paciente: ");
        String dni = scanner.nextLine().trim().toUpperCase();
        System.out.print("Fecha de Nacimiento (AAAA-MM-DD): ");
        LocalDate fechaNac = LocalDate.parse(scanner.nextLine().trim());

        // Pedimos datos Historia
        System.out.print("Nro. de Historia (ej: HC-1001): ");
        String nroHistoria = scanner.nextLine().trim().toUpperCase();
        System.out.print("Grupo Sanguíneo (A+, O-, AB+, etc.): ");
        GrupoSanguineo gs = GrupoSanguineo.fromValue(scanner.nextLine().trim().toUpperCase());
        System.out.print("Antecedentes (opcional): ");
        String antecedentes = scanner.nextLine().trim();
        System.out.print("Medicación Actual (opcional): ");
        String medicacion = scanner.nextLine().trim();
        System.out.print("Observaciones (opcional): ");
        String observaciones = scanner.nextLine().trim();

        // Creamos los objetos
        HistoriaClinica nuevaHistoria = new HistoriaClinica(null, false, nroHistoria, gs, antecedentes, medicacion,
        Paciente nuevoPaciente = new Paciente(null, false, nombre, apellido, dni, fechaNac, nuevaHistoria); // Asoci

        // Llamamos al SERVICIO transaccional
        // (El método se llama 'insertar' en la nueva plantilla)
```

5.2. Método insertar en PacienteService.java

```java
public void insertar(Paciente paciente) throws Exception {
    validatePaciente(paciente);

    HistoriaClinica historia = paciente.getHistoriaClinica();
    if (historia == null) {
        throw new IllegalArgumentException("El paciente debe tener una historia clinica para ser insertado");
    }

    Connection conn = null;
    try {
        conn = DatabaseConnection.getConnection();
        conn.setAutoCommit(false);

        // Validación de unicidad dentro de la transaccion
        validateDniUnique(paciente.getDni(), null, conn);
        if (historiaClinicaDao.getByNroHistoria(historia.getNroHistoria(), conn)!= null) {
            throw new IllegalArgumentException("Error: Ya existe una historia clínica con el Nro " + historia.getNro
        }
        //Creamos La hisotira clinica
        historiaClinicaDao.crear(historia,conn);
        //Creamos el paciente
        pacienteDao.crear(paciente, conn);

        conn.commit();
    } catch (Exception e) {
        if (conn != null) conn.rollback();
        throw new SQLException("Error al insertar paciente con Historia: " + e.getMessage(), e);
    } finally {
        if (conn != null) {
            conn.setAutoCommit(true);
            conn.close();
```

6. Conclusiones

- Se logró implementar un sistema funcional para la gestión de pacientes e historias clínicas con persistencia segura y manejo transaccional.
- El diseño modular y en capas facilita el mantenimiento y futuras ampliaciones.
- La interfaz por consola es simple pero efectiva para la interacción del usuario.
- Se recomienda implementar pruebas unitarias y validar con más casos extremos.