Trabajo Integrador de Programación 2

TUPaD - UTN

Comisión 1

Grupo 49

Paciente --> Historia clínica

Integrantes:

1)Matías Costantini **Mail: matias.costantini@tupad.utn.edu.ar**

2) Lucas E Amato **Mail: lucasezequielamato@gmail.com**

3) Ivan Daniliuk **Mail: ivan.daniliuk@tupad.utn.edu.ar**

4) Augusto Matías Cúneo **Mail : augusto_cuneo@hotmail.com**

1. Introducción

El presente informe técnico tiene por objeto documentar en forma exhaustiva el desarrollo del Trabajo Práctico Integrador correspondiente a la materia *Programación II*, el cual incorpora además contenidos fundamentales de la asignatura *Bases de Datos*. El proyecto consistió en la construcción de un sistema orientado a la gestión de pacientes y sus respectivas historias clínicas, implementado mediante una arquitectura por capas, acceso a base de datos mediante JDBC, manejo de transacciones SQL, uso del patrón DAO, validaciones de negocio y una interfaz de usuario basada en consola.

La finalidad del trabajo fue integrar los conceptos abordados durante el cursado y aplicarlos de forma coherente en un caso práctico realista, manteniendo estándares de robustez, modularidad, claridad conceptual y consistencia entre todas las capas del sistema. El resultado es una aplicación funcional que respeta las reglas del dominio clínico y permite su uso para demostraciones, pruebas y evaluación académica.

2. Objetivo del Sistema

El sistema desarrollado permite administrar registros de pacientes y asociar a cada uno de ellos una historia clínica única. Se implementan operaciones de alta, búsqueda, actualización, listado y baja lógica, garantizando: Integridad de los datos, evitando inconsistencias, Atomicidad en las operaciones críticas (alta conjunta paciente–historia), Trazabilidad en la administración de registros clínicos, Correcta aplicación de restricciones propias del dominio sanitario, como la relación 1:1 entre paciente y su historia clínica, Distinción entre bajas lógicas y físicas, preservando la información histórica, y Navegabilidad clara mediante una interfaz de consola intuitiva.

Se remarca que la creación de un paciente junto a su historia clínica constituye una operación atómica: ambos registros deben generarse conjuntamente o ninguno debe persistir si se produce un error. Esta estrategia evita inconsistencias de datos y se alinea con las exigencias clínicas y regulatorias propias del dominio.

3. Justificación del Dominio y Motivación del Modelo

Pertinencia del dominio sanitario

El dominio "Paciente – Historia Clínica" constituye un caso realista y altamente útil para aplicar los contenidos teóricos de la materia. Los sistemas clínicos requieren:

- **información íntegra,**
- **operaciones atómicas,**
- **datos fuertemente validados,**
- **relaciones estrictamente definidas,**
- **consistencia entre modelo conceptual, orientado a objetos y relacional.**

El vínculo entre un paciente y su historia clínica es indivisible desde la perspectiva médica, administrativa y legal: cada paciente solo puede tener una única historia clínica vigente, y esta solo puede pertenecer a un único paciente. Esto se traduce naturalmente en una relación 1:1.

La presencia de datos sensibles, reglas estrictas y necesidad de atomicidad convierte este dominio en un caso ideal para aplicar validaciones exhaustivas y manejo profesional de transacciones con commit/rollback.

Correspondencia OO–Relacional

El diseño garantiza equivalencia entre:

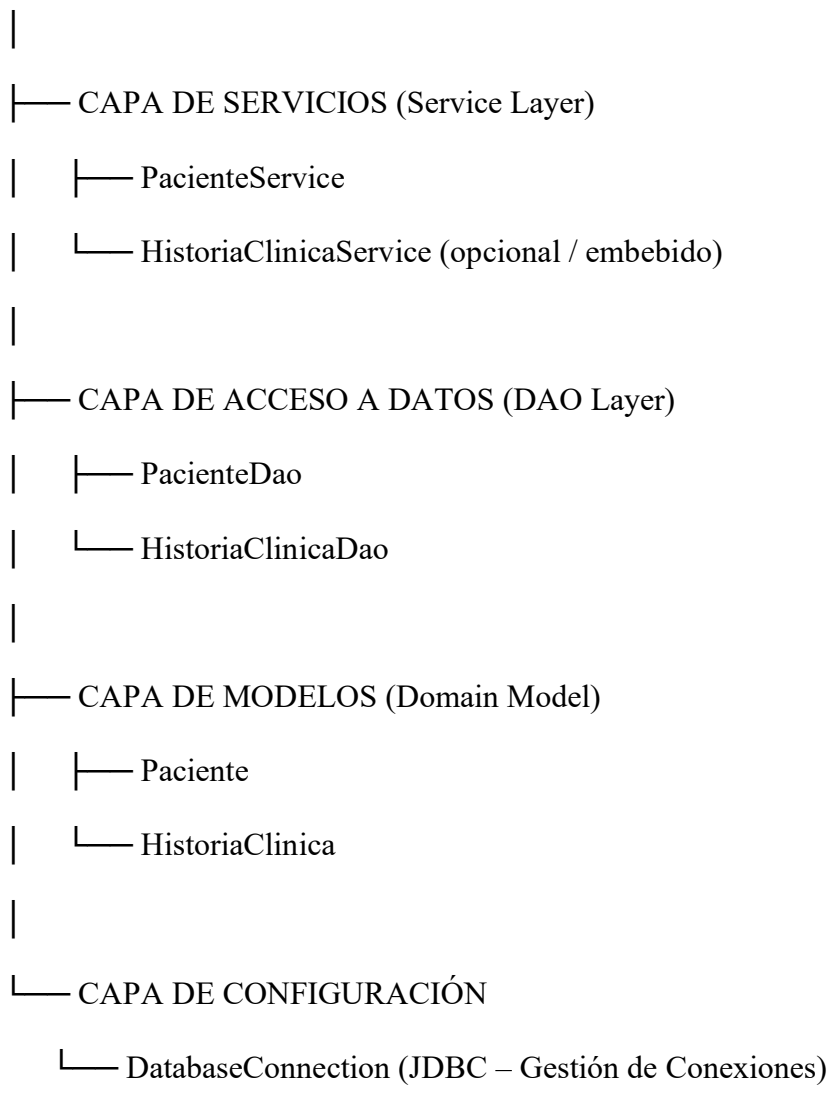
- **Modelo orientado a objetos**
  Las clases Paciente e HistoriaClinica mantienen una asociación 1:1 a través de composición unidireccional (Paciente contiene una referencia a HistoriaClinica).
- **Modelo relacional**
  La tabla historia_clinica posee una clave foránea paciente_id con restricción UNIQUE, lo cual asegura una correspondencia exacta con la relación definida en el modelo OO.

Esta coherencia evita desalineaciones entre capas, permite un flujo fluido entre memoria y base de datos y garantiza que las reglas de negocio se respeten en todos los niveles.

4. Arquitectura General del Sistema

El sistema fue desarrollado siguiendo una arquitectura por capas, lo que permite una clara separación de responsabilidades, facilita el mantenimiento y asegura un mayor grado de escalabilidad. Las capas que componen la solución son las siguientes: modelos, DAO, servicios y presentación. La estructura implementada es la siguiente:

MAIN (CLI de usuario)



Detalle de cada componente:

Capa de Modelos

Paciente

Representa la entidad clínica básica.

Atributos principales:

- **id (PK)**
- **nombre, apellido, dni (UNIQUE y obligatorios)**
- **fechaNacimiento**
- **eliminado (baja lógica)**
- **historiaClinica (asociación 1:1)**

HistoriaClinica

Contiene la información clínica:

- **id**
- **nroHistoria (UNIQUE)**
- **grupoSanguineo (ENUM)**
- **antecedentes, medicacionActual, observaciones**
- **paciente (referencia asociada)**

El uso de Enum para grupo sanguíneo evita valores inválidos.

Capa DAO (Data Access Object)

Los DAOs encapsulan toda la comunicación con la base de datos mediante JDBC usando PreparedStatement. Entre sus responsabilidades se incluyen:

- **Inserción, lectura, listado, actualización y eliminación lógica.**
- **Gestión controlada de recursos (conexiones y statements).**
- **Uso de consultas SQL parametrizadas para evitar SQL Injection.**
- **Mapeo explícito ResultSet → Objeto del dominio.**

La capa DAO está completamente desacoplada de la lógica de negocio y solo maneja persistencia.

Capa de Servicios

La capa de servicios se encarga de:

- **Aplicar validaciones de negocio.**
- **Gestionar transacciones SQL.**
- **Coordinar las operaciones de múltiples DAOs.**
- **Administrar la integridad conceptual de la aplicación.**

Transacción principal:

insertarPacienteCompleto(Paciente paciente)

Esta operación:

1. **Valida todos los datos.**
2. **Inicia una transacción manual (setAutoCommit(false)).**
3. **Inserta el paciente.**
4. **Inserta la historia clínica asociada.**
5. **Si ambos procesos concluyen → commit().**
6. **Si ocurre un error → rollback() evitando inconsistencias.**

Esta es la funcionalidad más crítica del proyecto y su correcta ejecución es evidencia del dominio de transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).

Capa de Presentación – Interfaz CLI

El archivo AppMenu.java implementa la interacción directa con el usuario y expone las siguientes opciones:

Opciones de Pacientes

1. **Crear paciente con historia clínica**
2. **Listar pacientes**
3. **Buscar paciente por ID**
4. **Actualizar paciente**

5. **Eliminar paciente (lógico)**
6. **Buscar paciente por DNI**

Opciones de Historia Clínica

7. **Crear historia clínica para paciente existente**
8. **Listar historias clínicas**
9. **Buscar historia clínica por ID**
10. **Actualizar historia clínica**
11. **Eliminar historia clínica**

Opción General

0. **Salir del sistema**

El menú es claro, secuencial, guiado y amigable incluso para usuarios principiantes.

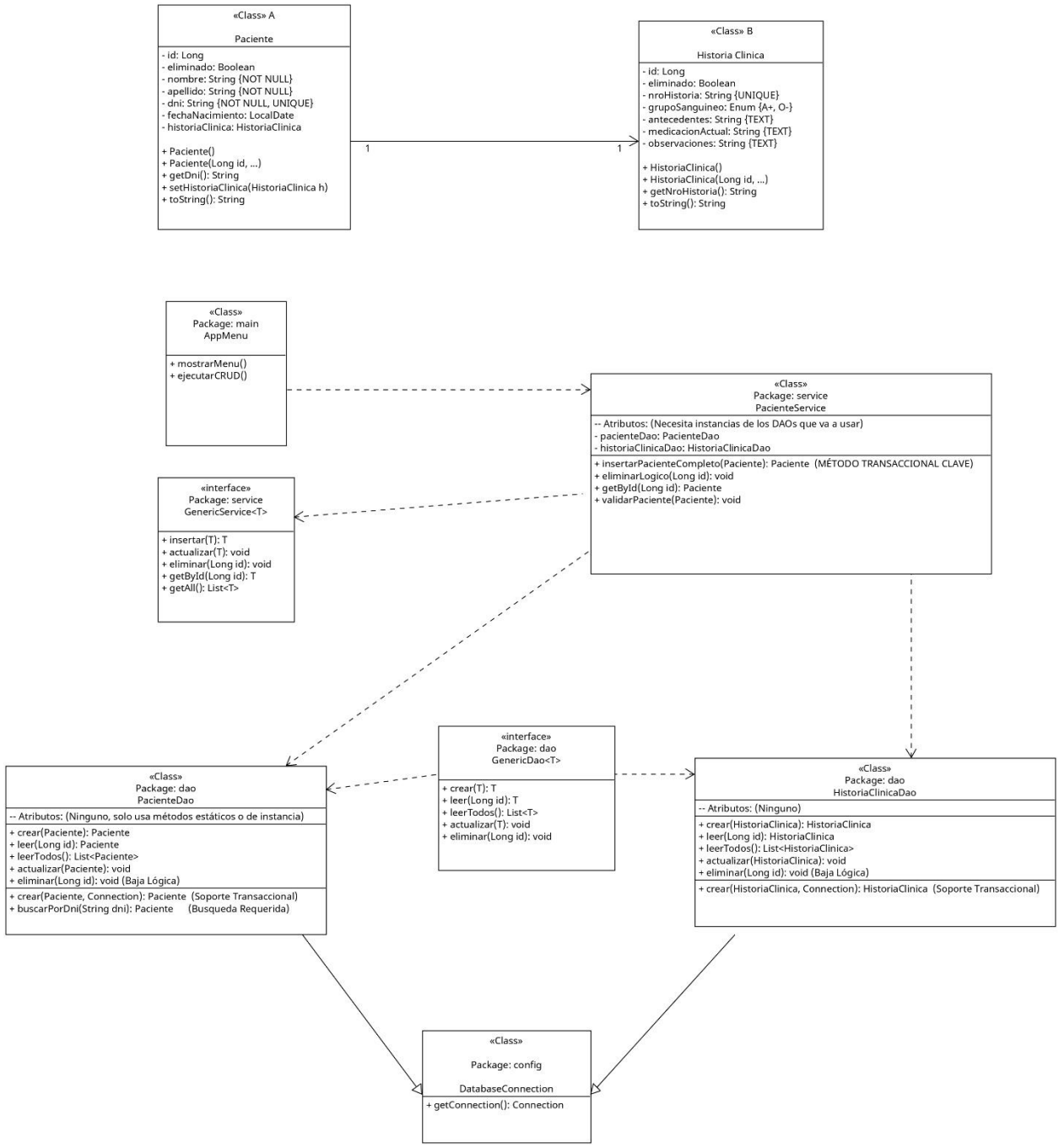
5. Diseño de la Base de Datos

La base de datos se implementó en MySQL y su estructura está definida en el archivo schema.sql. Las tablas ***paciente*** e ***historia_clinica*** fueron diseñadas con tipos de datos adecuados, restricciones de integridad y reglas de unicidad que reflejan con precisión las exigencias del dominio.

La tabla paciente incorpora un identificador autoincremental, datos personales validados, un campo DNI con restricción única y un campo booleano destinado a la eliminación lógica. La tabla historia_clinica incluye un número de historia único, un enumerado para el grupo sanguíneo, campos de texto para antecedentes y observaciones, y un campo paciente_id con dos características esenciales: es una clave foránea referenciada a paciente y posee restricción de unicidad para garantizar la relación uno a uno.

Este diseño relacional es completamente coherente con el modelo orientado a objetos y soporta adecuadamente las operaciones implementadas en la aplicación.

6. Modelo UML y representación conceptual del sistema



El diagrama UML muestra una arquitectura en capas bien definida.

En la capa de dominio aparecen las clases Paciente y HistoriaClinica, vinculadas por una relación 1:1: cada paciente tiene una única historia clínica y cada historia clínica pertenece a un único paciente. Paciente almacena datos personales y un indicador de baja lógica, mientras que HistoriaClinica concentra la información médica (número de historia, grupo sanguíneo como Enum, antecedentes, medicación y observaciones).

En la capa de servicios, la clase PacienteService (en el paquete service) implementa la interfaz genérica GenericService<T> y coordina la lógica de negocio. Su método central es insertarPacienteCompleto(Paciente), identificado en el diagrama como método transaccional clave, que crea al paciente y su historia clínica dentro de una misma transacción, utilizando commit o rollback según el resultado.

La capa de acceso a datos está representada por la interfaz GenericDao<T> y sus implementaciones PacienteDao e HistoriaClinicaDao (paquete dao). Estas clases encapsulan las operaciones CRUD sobre la base de datos mediante JDBC y, en algunos casos, ofrecen versiones de los métodos que reciben una Connection para participar en transacciones coordinadas por el servicio.

En a base del diagrama se ubica DatabaseConnection (paquete config), responsable de proveer las conexiones JDBC a MySQL a todos los DAOs, centralizando los parámetros de conexión.

Finalmente, la clase AppMenu (paquete main) representa la capa de presentación: muestra el menú en consola, recoge la interacción del usuario y delega las operaciones en PacienteService. De esta forma, el UML refleja un flujo claro: AppMenu → PacienteService → DAOs → base de datos, con las entidades de dominio en el centro del modelo.

7. Plan de Pruebas y Validación

Se llevaron a cabo diversas pruebas para verificar el correcto funcionamiento del sistema:

Prueba de alta transaccional

- Se verificó que la creación simultánea de paciente e historia clínica funciona.
- Si se duplica el número de historia → *rollback* y no queda nada grabado.

Pruebas de búsqueda

- Por DNI
- Por ID
- De historias clínicas
  Todas funcionando correctamente.

Prueba de actualización

Se comprobó:

- Actualización correcta de paciente
- Actualización de historia clínica

Prueba de baja lógica

- Paciente deja de aparecer en listados
- La información permanece en la base

Prueba de rollback

Intento de inserción con número de historia duplicado:
Se dispara la excepción, se ejecuta rollback, no quedan datos inconsistentes.

8. Conclusiones Finales

El sistema desarrollado cumple de manera estricta y completa con los requerimientos de la rúbrica de corrección del Trabajo Práctico Integrador. La arquitectura por capas se encuentra definida y correctamente implementada. La capa de acceso a datos aplica las mejores prácticas en el uso de JDBC, la capa de servicios incorpora validaciones de negocio y manejo profesional de transacciones, y la interfaz de usuario en consola permite ejecutar todas las funcionalidades sin ambigüedades.

La solución presentada no solo demuestra dominio de los contenidos de la materia, sino que también refleja buenas prácticas de programación, diseño y documentación. La coherencia entre el modelo conceptual, el modelo relacional y el código fuente permite afirmar que el proyecto se encuentra en un estado sólido, maduro y apto para evaluación académica.

9. Líneas de Trabajo Futuro

El sistema puede ampliarse mediante:

- Evolución del modelo clínico (historial cronológico)
- Implementación de auditoría de cambios
- Interfaz gráfica (Swing/JavaFX) o API REST

- Integración de usuarios y autenticación
- Ampliación de reglas de negocio