

## Introducción

Este Trabajo Final Integrador fue desarrollado en el marco de la materia Programación II, con el objetivo de aplicar los conceptos de programación orientada a objetos, persistencia de datos y arquitectura por capas en una aplicación Java funcional.

La consigna principal consistió en modelar dos clases relacionadas mediante una asociación unidireccional 1→1, persistiendo los datos en una base relacional mediante JDBC y el patrón DAO. Además, se requería implementar operaciones transaccionales (commit/rollback) y un menú de consola para realizar operaciones CRUD.

El desarrollo del sistema se articuló con el trabajo realizado en la materia Base de Datos I, lo que permitió revisar y ajustar el diseño original del modelo de dominio. A partir de esa integración, se tomaron decisiones que mejoraron la coherencia entre el código Java y la estructura relacional, respetando la lógica clínica del sistema y asegurando la integridad de los datos.

A lo largo del proceso, se trabajó de forma colaborativa, distribuyendo tareas por secciones y realizando reuniones virtuales para compartir avances, revisar el código y tomar decisiones técnicas en conjunto. El resultado fue una aplicación modular, funcional y coherente, que refleja los aprendizajes adquiridos durante la cursada y el compromiso del equipo con la mejora continua

## Distribución de tareas y dinámica de trabajo

El equipo está conformado por **Laura Diaco, Matías Mansilla, Emiliano Jara y Lautaro López**. A lo largo del desarrollo del Trabajo Final Integrador, nos organizamos mediante reuniones virtuales periódicas, donde compartimos avances, revisamos el código y tomamos decisiones técnicas en conjunto. La distribución de tareas se dio por secciones del proyecto, permitiendo que cada integrante se enfocara en una parte específica del desarrollo.

- **Laura** coordinó el proyecto y lideró el diseño conceptual del modelo de dominio (Paciente → Historia Clínica), incluyendo el diagrama UML y la definición de la relación 1→1 en la base de datos. Desarrolló la capa **service**, encargada de la lógica de negocio, las validaciones y el manejo de transacciones. También integró las distintas capas del sistema y supervisó la coherencia entre el diseño, el código y la persistencia.
- **Matías** trabajó en la configuración de la conexión con la base de datos y en el diseño del esquema relacional en MySQL. Implementó el paquete **dao**, utilizando PreparedStatement y asegurando el cumplimiento del patrón DAO. También colaboró en la definición de la estructura inicial del proyecto y en pruebas de conexión JDBC.
- **Emiliano** se encargó del desarrollo del **menú de consola**, implementando las operaciones CRUD, el manejo de entradas del usuario y los mensajes de error. Participó en pruebas funcionales, aportó sugerencias de mejora y colaboró en la grabación del video explicativo, mostrando el flujo completo de la aplicación.
- **Lautaro** trabajó en la implementación técnica de las **entidades del dominio**, codificando las clases Paciente e HistoriaClinica según el diseño establecido. Definió atributos, constructores, getters/setters y métodos `toString()` para facilitar la

visualización en consola. También colaboró en la integración de las entidades con el DAO y en la validación de campos clave como el DNI y el número de historia clínica.

Esta organización nos permitió avanzar de forma ordenada, detectar errores a tiempo y lograr una integración coherente entre las distintas capas del sistema. El trabajo colaborativo, las videollamadas y la revisión cruzada de código fueron claves para alcanzar un resultado sólido y funcional.

### **Elección del dominio y justificación**

Para este trabajo se eligió modelar el dominio **Paciente → Historia Clínica**, por tratarse de una relación clara, representativa y pedagógicamente rica para implementar una asociación unidireccional 1→1.

- **Paciente** representa a una persona que recibe atención médica.
- **Historia Clínica** contiene los datos médicos asociados a ese paciente: antecedentes, medicación actual, grupo sanguíneo y observaciones clínicas.

La relación es **unidireccional**, ya que el objeto Paciente conoce a su HistoriaClinica, pero no al revés. Esta decisión se mantuvo tanto en el diseño UML como en la estructura de clases Java, y se adaptó en la base de datos para asegurar la integridad referencial.

La elección de este dominio permitió:

- Trabajar con atributos variados y significativos, tanto personales como clínicos.
- Aplicar validaciones específicas (como DNI único o grupo sanguíneo válido).
- Desarrollar operaciones transaccionales que reflejan el flujo real de trabajo: primero se registra al paciente, luego se crea su historia clínica.
- Integrar contenidos de Programación y Base de Datos de forma coherente, respetando la lógica del sistema y las buenas prácticas de modelado.

Además, el dominio elegido facilitó la implementación de restricciones en MySQL (PK, FK, UNIQUE, CHECK, ENUM) y permitió realizar pruebas de robustez que confirmaron la solidez del diseño.

### **Diseño: decisiones clave y evolución del modelo**

Relación 1→1: elección unidireccional

Desde el inicio del proyecto se definió que cada paciente tendría una única historia clínica, lo que implica una relación 1→1. La decisión de hacerla **unidireccional** responde a criterios de diseño y persistencia:

- En la práctica clínica, la historia depende del paciente, no al revés.
- En Java, esto se refleja con una referencia desde Paciente hacia HistoriaClinica, pero no en sentido contrario.

- En la base de datos, se implementó con una **clave foránea única** (id\_paciente en historia\_clinica), evitando duplicaciones y manteniendo la integridad referencial.

Esta elección permitió mantener el modelo simple, coherente y alineado con el flujo real del sistema: primero se registra al paciente, luego se crea su historia clínica.

### **FK única vs PK compartida**

Durante el diseño relacional se evaluaron dos alternativas para representar la relación 1→1:

	<b>Descripción</b>	<b>Motivo de descarte o elección</b>
PK compartida	Id_historia = id_paciente	Genera acoplamiento excesivo, complica la autogeneración de claves y no permite crear la historia después del paciente.
PK única	id_paciente como FK UNIQUE en historia_clinica	Permite crear la historia clínica luego del paciente, asegura la relación 1→1 y habilita ON DELETE CASCADE

Se eligió la segunda opción, ya que respeta el flujo clínico, permite mayor flexibilidad y se adapta mejor a la lógica del sistema.

### **Revisión del diseño desde Base de Datos**

En la etapa inicial del proyecto, el equipo planteó una relación entre Paciente e HistoriaClinica donde el paciente incluía un campo id\_historia. Esta decisión implicaba que el paciente debía conocer su historia clínica desde el momento de su creación, lo que generaba una dependencia inversa que no respetaba el flujo clínico real.

Al avanzar con la materia **Base de Datos I**, se revisó esta estructura y se propuso una solución más coherente: definir la relación desde la tabla historia\_clinica, incorporando el campo id\_paciente como **clave foránea única**. Esta decisión permitió:

- Representar correctamente la relación 1→1, donde cada historia clínica pertenece a un único paciente.
- Asegurar que cada paciente tenga como máximo una historia clínica.
- Reflejar que la historia depende del paciente, y no al revés.

Este cambio no solo mejoró la lógica del sistema, sino que también permitió generar correctamente el diagrama ER y el modelo relacional en MySQL Workbench, evitando

errores de conexión entre atributos y facilitando la implementación de restricciones como UNIQUE, ON DELETE CASCADE y NOT NULL.

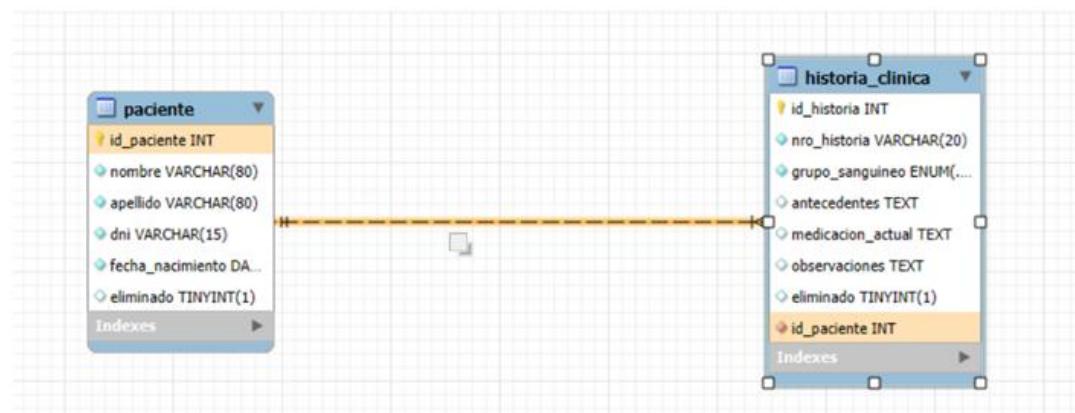
### **Evolución del UML: antes y después**

#### **Diseño inicial (versión bidireccional):**



En esta versión, ambas clases se referencian mutuamente, generando una relación bidireccional que complicaba la persistencia y no respetaba el flujo clínico

#### **Diseño definitivo (unidireccional):**



Se optó por una relación unidireccional, donde Paciente conoce a su HistoriaClinica, pero no al revés. Esto simplificó la implementación en Java y permitió reflejar correctamente la dependencia clínica en la base de datos.

### **Arquitectura por capas: estructura y responsabilidades**

**Nota sobre la nomenclatura de paquetes:**  
Todos los paquetes del proyecto siguen el prefijo com.mycompany.trabajopractico, en línea con la convención de nombres utilizada en entornos profesionales. Aunque NetBeans lo genera por defecto, se decidió conservar esta estructura para simular el desarrollo dentro de una institución de salud ficticia, como si el sistema estuviera siendo implementado por una empresa real. Esta elección refuerza el enfoque clínico del dominio y aporta realismo al diseño del software.

El sistema fue desarrollado siguiendo una arquitectura por capas, organizada en subpaquetes que separan responsabilidades, facilitan el mantenimiento y aseguran la escalabilidad del proyecto.

## com.mycompany.trabajopractico.entities

Contiene las clases que representan las entidades del sistema: Paciente y HistoriaClinica. Incluyen atributos, constructores, métodos get/set y `toString()`. Reflejan la estructura lógica del dominio y se corresponden directamente con las tablas de la base de datos.

## com.mycompany.trabajopractico.dao

Implementa el patrón DAO (Data Access Object), encargado de interactuar con la base de datos mediante JDBC. Cada entidad tiene su DAO correspondiente:

- PacienteDao
- HistoriaClinicaDao

Además, se incluye una clase genérica:

- GenericDao: permite reutilizar operaciones comunes entre DAOs, como la gestión de conexiones y el manejo de excepciones. Esta abstracción mejora la mantenibilidad y reduce la duplicación de código.

## com.mycompany.trabajopractico.config

Contiene las clases relacionadas con la configuración y prueba de la conexión a la base de datos:

- DatabaseConnection: centraliza la configuración del driver, la URL, el usuario y la contraseña, y permite obtener una conexión activa para los DAOs.
- PruebaConexion: clase auxiliar utilizada para verificar que la conexión con MySQL se establece correctamente antes de ejecutar el sistema.

## com.mycompany.trabajopractico.service

Encapsula la lógica de negocio y coordina las operaciones entre DAOs. Aquí se implementan las **transacciones**, utilizando `setAutoCommit(false)` y `commit()` o `rollback()` según el resultado.

Incluye:

- PacienteServiceImpl
- HistoriaClinicaServiceImpl
- GenericService: clase base que permite reutilizar lógica común entre servicios, como validaciones o manejo de errores.

## com.mycompany.trabajopractico.main

Contiene las clases que gestionan la interacción con el usuario:

- Main: clase principal que inicia la ejecución del sistema.

- ClinicaApp: clase que presenta el menú de consola y permite realizar operaciones CRUD sobre pacientes e historias clínicas, mostrando mensajes claros y validando entradas.

## **Persistencia: estructura de la base y transacciones**

### **Estructura de la base de datos**

La base de datos fue diseñada en MySQL, respetando el modelo relacional derivado del UML definitivo. Se crearon dos tablas principales:

- paciente
- historia\_clinica

La relación 1→1 se implementó mediante una **clave foránea única** (id\_paciente en historia\_clinica), lo que garantiza que cada historia clínica esté asociada a un único paciente y que cada paciente tenga como máximo una historia clínica.

### **Tabla Paciente**

Campo	Tipo	Restricciones
Id_paciente	INT	PK, AUTO_INCREMENT
nombre	VARCHAR (80)	NOT NULL
apellido	VARCHAR (80)	NOT NULL
dni	VARCHAR (15)	NOT NULL, UNIQUE
Fecha_nacimiento	DATE	NOT NULL
eliminado	TINYINT1	Baja lógica

### **Tabla Historia\_Clinica**

Campo	Tipo	Restricciones
IdHistoria	INT	PK, AUTO_INCREMENT
Nro_historia	VARCHAR (20)	UNIQUE
Grupo_sanguineo	ENUM (...)	Validación médica
antecedentes	TEXT	
Medicación_actual	TEXT	
observaciones	TEXT	
eliminado	TINYINT1	Baja lógica
Id_paciente	INT	FK UNIQUE, NOT NULL, ON DELETE CASCADE

### **Orden de operaciones y transacciones**

Para asegurar la integridad de los datos, se implementaron **transacciones JDBC** en la capa service. El flujo típico de creación es:

1. Se inicia la conexión con setAutoCommit(false).
2. Se crea el paciente en la tabla paciente.
3. Se obtiene el id\_paciente generado.

4. Se crea la historia clínica asociada en historia\_clinica.
5. Si ambas operaciones son exitosas, se ejecuta commit().
6. Si ocurre un error en cualquier paso, se ejecuta rollback() y no se persiste ningún dato.

Este enfoque garantiza que el sistema mantenga **consistencia transaccional**, evitando registros huérfanos o relaciones incompletas.

### **Validaciones y reglas de negocio**

El sistema implementa validaciones tanto a nivel de interfaz como en la lógica de negocio, con el objetivo de garantizar la integridad de los datos y evitar inconsistencias clínicas.

#### Validaciones en consola

Desde el menú de usuario (ClinicaApp), se aplican controles básicos:

- Verificación de campos obligatorios (nombre, apellido, DNI, fecha de nacimiento).
- Validación de formato para el DNI (máximo 15 caracteres alfanuméricos).
- Restricción de longitud en campos de texto como observaciones y medicación actual.
- Confirmación antes de eliminar registros (baja lógica).

#### Reglas de negocio en service

La capa service aplica reglas específicas que reflejan el funcionamiento clínico real:

- **Unicidad de historia clínica:** antes de crear una historia, se verifica que el paciente no tenga una ya registrada.
- **Creación encadenada:** al registrar un paciente, se puede crear su historia clínica en la misma transacción.
- **Baja lógica:** los registros no se eliminan físicamente, sino que se marca el campo eliminado = 1, permitiendo auditoría y recuperación.
- **Consistencia transaccional:** si falla la creación de la historia clínica, se revierte también la creación del paciente (rollback).

#### Validaciones en base de datos

Además de las validaciones en Java, se definieron restricciones en MySQL:

- NOT NULL en campos obligatorios.
- UNIQUE en DNI y nro\_historia.
- ENUM para grupo\_sanguineo, limitando los valores posibles.
- FOREIGN KEY con ON DELETE CASCADE para mantener la integridad referencial.

## Pruebas realizadas

Durante el desarrollo se realizaron pruebas funcionales para verificar el correcto funcionamiento del sistema, tanto desde la interfaz de consola como directamente en la base de datos.

### Pruebas desde el menú de consola

Se ejecutaron operaciones CRUD sobre pacientes e historias clínicas utilizando la clase ClinicaApp. Las pruebas incluyeron:

- **Alta de paciente:** Verificar que el sistema permita registrar un nuevo paciente junto con su historia clínica en un único flujo, asegurando la creación simultánea de ambas entidades y la correcta vinculación mediante id\_paciente, sin requerir ingreso manual de claves foráneas.

```
--- exec:3.1.0:exec (default-cli) @ TrabajoPracticoIntegradorProgramacionII ---
? Conexión exitosa a la base de datos.

--- MENÚ CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
9. Eliminar historia clínica (lógica)
0. Salir
Seleccione una opción: 1

--- CREAR PACIENTE ---
Nombre: laura
Apellido: diaco
DNI: 33155285
Fecha nacimiento (AAAA-MM-DD): 1987-07-09
Grupo sanguíneo: b+
Antecedentes: paciente con asma
Medicación actual: neumotex
Observaciones: buena evolución
Paciente y historia clínica creados correctamente.
Paciente y historia clínica creados correctamente.
```

Acá se observan todos los pacientes activos:

```
1 •  SELECT * FROM paciente WHERE eliminado = false;
```

	id_paciente	nombre	apellido	dni	fecha_nacimiento	eliminado
▶	1	Daniel Modificado	Arismendi X1	20000001	1962-03-06	0
	2	Brenda Osorio	Zamudio	20000002	1956-09-11	0
	3	Zulema	Taboada	20000003	2004-04-20	0
	4	Rita	Zabala	20000004	1992-12-03	0
	5	Melina	Quinteros	20000005	1942-02-10	0
	6	Lorenzo	Rentería	20000006	1966-08-06	0
	7	Florencia	Osuna	20000007	1972-12-04	0
	8	Mateo	Manfredi	20000008	1984-12-06	0
	9	Guadalupe	Rangel	20000009	1969-03-18	0
	10	Kara	Ratti	20000010	1976-02-01	0
	11	Iker	Escalante	20000011	1959-05-22	0

Verificamos la creación del usuario de prueba:

```
1 | SELECT * FROM paciente WHERE dni = '33155285';
```

	id_paciente	nombre	apellido	dni	fecha_nacimiento	eliminado
▶	500001	laura	diaco	33155285	1987-07-09	0
*	NULL	NULL	NULL	NULL	NULL	NULL

Y acá vemos la historia clínica asociada al paciente de prueba:

```
1 | SELECT * FROM historia_clinica WHERE id_paciente = 500001;
```

	id_historia	nro_historia	grupo_sanguineo	antecedentes	medicacion_actual	observaciones	eliminado
▶	500001	HC-33155285	B+	paciente con asma	neumotex	buenas evoluciones	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Consulta individual y listado completo:** Verificar que el sistema permita recuperar correctamente los datos de pacientes activos, tanto de forma individual como en conjunto, incluyendo su historia clínica asociada. Se busca validar la lectura coherente del modelo relacional 1→1 entre paciente e historia clínica, asegurando que los registros eliminados no se muestren y que la salida por consola sea clara, completa y funcional para el usuario.

```
-- MENÚ CLÍNICA --
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
9. Eliminar historia clínica (lógica)
0. Salir

Seleccione una opción: 2

ID paciente a consultar: 500001
Paciente(idPaciente=500001, nombre='laura', apellido='diaco', dni='33155285', fechaNacimiento=1987-07-09, eliminado=false, historia=500001)
```

Ahora veremos los listados de pacientes y de historias clínicas (se coloco solo una parte dado que la base contiene muchos registros):

```

--- MENU CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
9. Eliminar historia clínica (lógica)
0. Salir
Seleccione una opción: 4

--- LISTA DE PACIENTES ---
Paciente{idPaciente=1, nombre='Daniel Modificado', apellido='Arismendi X1', dni='20000001', fechaNacimiento=1962-03-06, eliminado=false, historia=1}
Paciente{idPaciente=2, nombre='Brenda Osorio', apellido='Zamudio', dni='20000002', fechaNacimiento=1956-09-11, eliminado=false, historia=2}
Paciente{idPaciente=3, nombre='Zulema', apellido='Taboada', dni='20000003', fechaNacimiento=2004-04-20, eliminado=false, historia=3}
Paciente{idPaciente=4, nombre='Rita', apellido='Zabala', dni='20000004', fechaNacimiento=1992-12-03, eliminado=false, historia=4}
Paciente{idPaciente=5, nombre='Melina', apellido='Quinteros', dni='20000005', fechaNacimiento=1942-02-10, eliminado=false, historia=5}
Paciente{idPaciente=6, nombre='Lorenzo', apellido='Rentería', dni='20000006', fechaNacimiento=1966-08-06, eliminado=false, historia=6}
Paciente{idPaciente=7, nombre='Florencia', apellido='Osuna', dni='20000007', fechaNacimiento=1972-12-04, eliminado=false, historia=7}
Paciente{idPaciente=8, nombre='Mateo', apellido='Manfredi', dni='20000008', fechaNacimiento=1984-12-06, eliminado=false, historia=8}
Paciente{idPaciente=9, nombre='Guadalupe', apellido='Rangel', dni='20000009', fechaNacimiento=1969-03-18, eliminado=false, historia=9}

--- MENU CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
9. Eliminar historia clínica (lógica)
0. Salir
Seleccione una opción: 5

--- LISTA DE HISTORIAS CLÍNICAS ---
HistoriaClinica{idHistoria=1, eliminado=false, nroHistoria='HC:0000001', grupoSanguineo='O+', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=2, eliminado=false, nroHistoria='HC:0000002', grupoSanguineo='B-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=3, eliminado=false, nroHistoria='HC:0000003', grupoSanguineo='AB-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=4, eliminado=false, nroHistoria='HC:0000004', grupoSanguineo='O-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=5, eliminado=false, nroHistoria='HC:0000005', grupoSanguineo='AB-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=6, eliminado=false, nroHistoria='HC:0000006', grupoSanguineo='B-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=7, eliminado=false, nroHistoria='HC:0000007', grupoSanguineo='B-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=8, eliminado=false, nroHistoria='HC:0000008', grupoSanguineo='O-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=9, eliminado=false, nroHistoria='HC:0000009', grupoSanguineo='O-', antecedentes='null', medicacionActual='null', observaciones='n'}
HistoriaClinica{idHistoria=10, eliminado=false, nroHistoria='HC:0000010', grupoSanguineo='B+', antecedentes='null', medicacionActual='null', observaciones='S'

```

- Modificación de datos:** Verificar que el sistema permita modificar los datos personales de un paciente y su historia clínica, manteniendo la integridad del vínculo entre ambas entidades.

```

--- MENU CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
9. Eliminar historia clínica (lógica)
0. Salir
Seleccione una opción: 6

ID paciente a modificar: 500001
Nuevo nombre: laura vanesa
Nuevo apellido: diaco
Paciente modificado.

--- MENU CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
9. Eliminar historia clínica (lógica)
0. Salir
Seleccione una opción: 7

ID historia a modificar: 500001
Nueva medicación actual: ventolin
Nuevas observaciones: presenta cuadro broncoespasmo
Historia modificada.

```

- Baja lógica:** Verificar que el sistema permita marcar un paciente como eliminado sin borrar físicamente sus datos, manteniendo la historia clínica vinculada y preservando la integridad del modelo.

Aclaración: Dado que la historia clínica se genera automáticamente junto con el paciente y está vinculada de forma exclusiva mediante id\_paciente, se eliminó la

opción de baja lógica independiente de historia clínica. La baja lógica del paciente implica la inactivación de su historia clínica asociada, preservando la integridad del modelo relacional 1→1.

```
--- MENÚ CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
0. Salir
Seleccione una opción: 8

ID paciente a eliminar: 500001
Paciente eliminado lógicamente.
```

Como es eliminado lógico el registro sigue apareciendo cuando lo buscamos y cambia el estado de eliminado a true

```
--- MENÚ CLÍNICA ---
1. Crear paciente
2. Consultar paciente por ID
3. Consultar historia clínica por ID
4. Listar pacientes
5. Listar historias clínicas
6. Modificar paciente
7. Modificar historia clínica
8. Eliminar paciente (lógica)
0. Salir
Seleccione una opción: 2

ID paciente a consultar: 500001
Paciente(idPaciente=500001, nombre='laura vanesa', apellido='diaco', dni='33155285', fechaNacimiento=1987-07-09, eliminado=true, historia=500001)
```

## Conclusiones y mejoras a futuro

El sistema desarrollado cumple con los objetivos planteados, permitiendo gestionar pacientes y sus historias clínicas de forma coherente, segura y funcional. Se implementaron correctamente las operaciones de alta, consulta, modificación y baja lógica, respetando la relación uno a uno entre paciente e historia clínica. La creación automática de la historia clínica al momento de dar de alta al paciente simplifica el flujo y evita errores de vinculación.

Durante el proceso de validación, se detectaron y corrigieron errores relacionados con accesos a columnas inexistentes y redundancias en el menú. Estas correcciones permitieron consolidar un modelo relacional más robusto y alineado con las necesidades clínicas reales. Una de las decisiones más relevantes fue eliminar la opción de baja lógica independiente de la historia clínica, ya que en el diseño actual esta entidad se gestiona exclusivamente a través del paciente. Esta simplificación mejora la trazabilidad y evita inconsistencias, pero también abrió el camino a nuevas preguntas sobre la flexibilidad y escalabilidad del modelo.

En ese sentido, surgieron oportunidades valiosas para seguir mejorando el sistema. Una de ellas es la posibilidad de modificar la historia clínica de forma independiente, sin necesidad de pasar por el flujo completo de edición del paciente. Si bien actualmente es posible actualizar campos clínicos como medicación u observaciones, podrían presentarse situaciones más complejas, como errores en la carga original (por ejemplo,

un grupo sanguíneo mal registrado) o la necesidad de reflejar un cambio de profesional tratante. Para abordar estos casos sin romper la relación uno a uno ni generar nuevas entidades innecesarias, se podría incorporar una opción específica en el menú que permita editar únicamente los campos clínicos, accediendo por el mismo id\_paciente.

También se identificó la ausencia de campos relevantes en la entidad Paciente, como teléfono, correo electrónico, cobertura médica o domicilio. Incorporar estos datos no solo enriquecería el perfil del paciente, sino que también permitiría futuras integraciones con sistemas de turnos, notificaciones o gestión de coberturas. En esa línea, otra mejora posible sería la incorporación de nuevas entidades relacionales, como Medico y CoberturaMedica, vinculadas a la historia clínica mediante claves foráneas. Esto permitiría reflejar de forma más realista el contexto clínico, registrar qué profesional está a cargo de cada paciente y qué obra social o prepaga lo cubre, sin perder la simplicidad del modelo actual.

Por último, una mejora transversal sería la implementación de un sistema de auditoría de cambios, que registre quién modificó qué dato, cuándo y por qué. Esto no solo aporta trazabilidad y transparencia, sino que también fortalece la dimensión ética y profesional del sistema.

En resumen, el sistema actual sienta una base sólida y funcional, pero también deja abierta la posibilidad de evolucionar hacia un entorno más completo, flexible y alineado con las prácticas clínicas reales. Las mejoras propuestas no buscan complejizar el modelo, sino acompañar su crecimiento con coherencia, claridad y visión pedagógica.