

## **Trabajo Final Integrador (TFI) - Programación II**

Grupo 5

Balaguer Joaquín

TUPaD, Universidad Tecnológica Nacional

Programación II

Ariel Enferrel

Federico Frankerberger

20 de noviembre de 2025

Informe del Sistema Empleado–Legajo .....	2
1. Elección del dominio y justificación .....	3
2. Diseño del sistema .....	3
2.1 Relación 1:1 .....	4
2.2 Estrategia de persistencia: FK única vs PK compartida .....	4
2.3 UML del sistema .....	5
3. Arquitectura por capas.....	5
3.1 Capa Config .....	6
3.2 Capa DAO .....	6
3.3 Capa Service .....	6
3.4 Capa Model.....	6
3.5 Capa Main.....	6
4. Persistencia .....	7
4.1 Estructura de la base de datos.....	7
4.2 Orden de operaciones.....	8
4.3 Manejo de transacciones .....	8
5. Validaciones y reglas de negocio .....	9
6. Conclusiones y mejoras futuras.....	9

# Informe del Sistema Empleado–Legajo

## 1. Elección del dominio y justificación

El dominio seleccionado para el desarrollo de esta aplicación es la **gestión de empleados**, con énfasis en la relación 1:1 entre un **Empleado** y su **Legajo**. Este dominio fue elegido por varias razones:

- Es un escenario frecuente en entornos corporativos y académicos, lo que facilita su comprensión y análisis.
- La relación 1:1 entre las entidades resulta ideal para explorar distintos enfoques de modelado.
- Permite demostrar un flujo completo de arquitectura en capas, desde presentación hasta persistencia.
- Es suficientemente simple para fines didácticos, pero con elementos reales de complejidad: integridad referencial, transacciones, validaciones y reglas de negocio.

En este contexto, **Empleado** representa a la persona dentro de la organización, mientras que **Legajo** contiene información administrativa vinculada de manera exclusiva a dicho empleado. Esta relación respalda funcionalidades que dependen de la correlación directa entre ambas entidades.

## 2. Diseño del sistema

El diseño partió de la identificación de las entidades principales y de su relación. Las decisiones clave incluyen:

### 2.1 Relación 1:1

Se determinó que cada **Empleado** debe poseer un único **Legajo**, y cada Legajo corresponde estrictamente a un solo Empleado. Esto implica restricciones en la base de datos y en el modelo del dominio.

### 2.2 Estrategia de persistencia: FK única vs PK compartida

Existen dos enfoques para modelar la relación 1:1:

- **Foreign Key única (FK única)**: la tabla hija contiene una columna que referencia a la PK de la tabla padre, con una restricción UNIQUE para asegurar la cardinalidad.
- **Primary Key compartida (PK compartida)**: ambas tablas comparten la misma clave primaria, garantizando una correspondencia estricta.

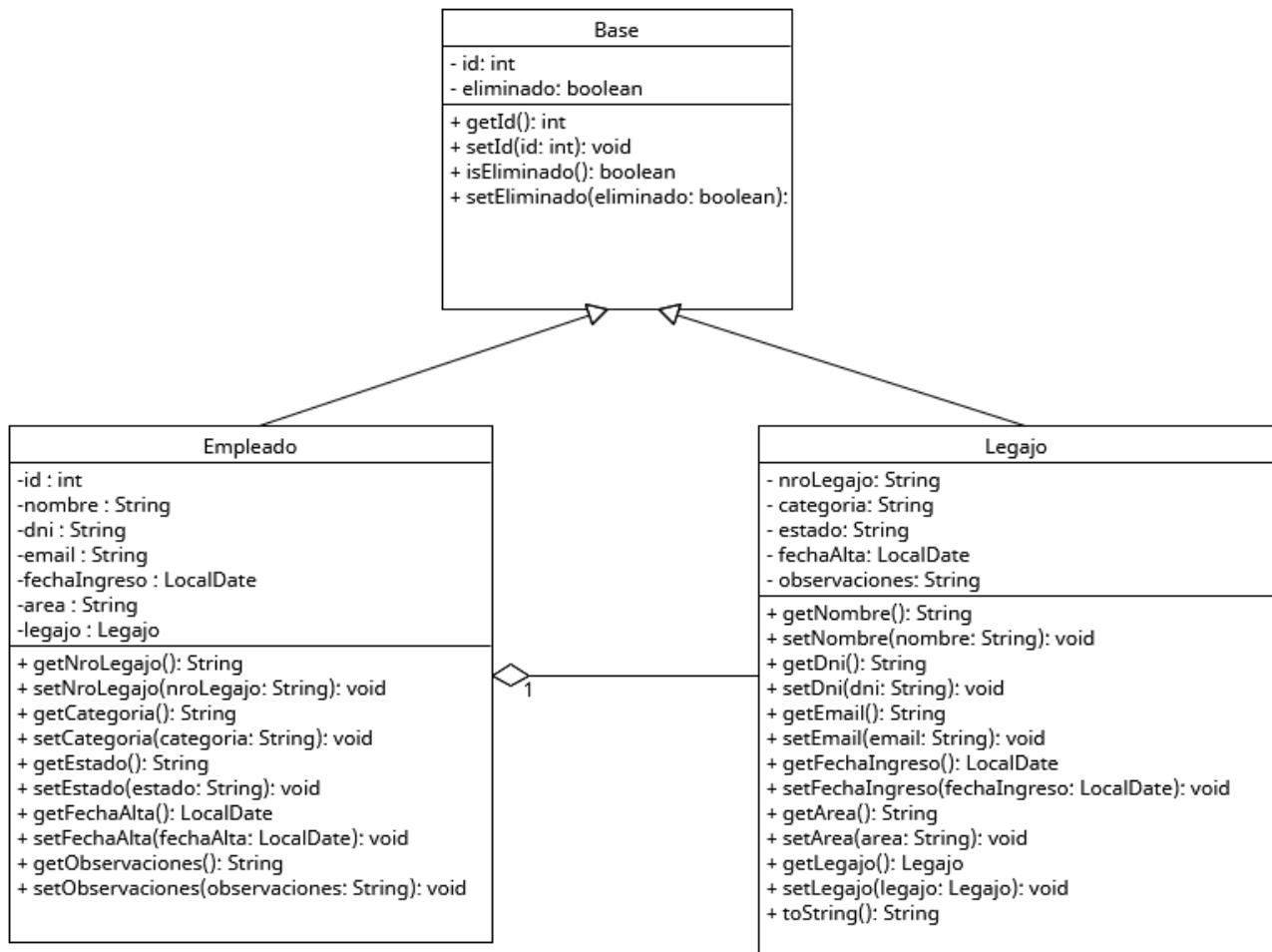
La aplicación utiliza la **FK única**, ya que:

- Simplifica la creación y manipulación de registros.
- Mantiene independencia entre las entidades, evitando acoplamientos rígidos.

- Permite operaciones CRUD más flexibles al no depender de claves primarias idénticas.

## 2.3 UML del sistema

El modelo conceptual:



## 3. Arquitectura por capas

La aplicación está organizada en una arquitectura multicapa, distribuyendo las responsabilidades en paquetes bien definidos.

### 3.1 Capa Config

- **DatabaseConnection**: administra conexiones JDBC.
- **TransactionManager**: controla transacciones, commits y rollbacks.

### 3.2 Capa DAO

- **GenericDAO**: operaciones CRUD genéricas.
- **EmpleadoDAO / LegajoDAO**: implementaciones específicas por entidad.

### 3.3 Capa Service

- **GenericService**: interfaz general de funcionalidades.
- **EmpleadoServiceImpl / LegajoServiceImpl**: validaciones, reglas de negocio, coordinación de DAOs.

### 3.4 Capa Model

- Contiene las entidades del dominio. Representa el modelo conceptual.

### 3.5 Capa Main

- Maneja la interacción en consola: menús, mensajes, flujo de usuario.

Esta separación permite:

- Reducir acoplamiento.
- Mejorar la mantenibilidad.
- Facilitar pruebas unitarias.

## 4. Persistencia

### 4.1 Estructura de la base de datos

Se consideran dos tablas principales:

#### Empleado

- id (PK)
- nombre
- apellido
- dni
- edad

#### Legajo

- id (PK)

- fechalIngreso
- puesto
- categoria
- id\_empleado (FK UNIQUE → Empleado.id)

## 4.2 Orden de operaciones

Debido a la relación 1→1, la creación de registros requiere un orden específico:

- **Alta:** primero Empleado → luego Legajo.
- **Baja:** primero Legajo → luego Empleado.
- **Actualización:** independiente, pero respetando integridad referencial.

## 4.3 Manejo de transacciones

Las transacciones se gestionan desde la capa Service utilizando el TransactionManager.

Se realizan:

- **commit** cuando todas las operaciones relacionadas (Empleado y Legajo) se completan correctamente.
- **rollback** si ocurre cualquier error en DAO o en lógica de negocio, garantizando consistencia.

El Service coordina múltiples DAOs dentro de una misma transacción, por ejemplo al crear ambos registros.

## 5. Validaciones y reglas de negocio

Se implementaron validaciones en el Service para asegurar la integridad de los datos:

- Validación de DNI: formato y unicidad.
- Validación de nombres: no nulos, con longitud mínima.
- Formato de fecha de ingreso en Legajo.

Reglas de negocio:

- Un empleado no puede tener más de un legajo.

Estas validaciones residen en el Service, no en DAO, para mantener separación adecuada de responsabilidades.

## 6. Conclusiones y mejoras futuras

El diseño implementado cumple con los objetivos del ejercicio y demuestra un ciclo completo de una arquitectura por capas, integrando entidades relacionadas 1:1, transacciones, validaciones y un menú interactivo.

Como mejoras futuras se sugieren:

- Implementar conexión mediante un pool (HikariCP).
- Migrar a JPA/Hibernate para reducir código manual.
- Añadir pruebas unitarias para DAOs y Services.
- Reemplazar el menú de consola por una interfaz gráfica o API REST.
- Mejorar el manejo de excepciones y mensajes al usuario.

El sistema, tal como está, ofrece una base sólida, clara y extensible que permite experimentar con distintos estilos de arquitectura y patrones de diseño, siendo adecuado para fines académicos y formativos.