Trabajo Final Integrador

Programación 2

Grupo 100

Alumnos:

- Matias Scalella
- Gabriel Osvaldo Roque
- Milagros Abril Airalde

Tecnicatura Universitaria en Programación (UTN)

Link al repositorio de GitHub: https://github.com/matiscalella/TFI_Grupo100_Scalella_Roque_Airalde_UTN

Link al video: https://youtu.be/iFxiG2wpDSc

DOMINIO

Para el desarrollo del Trabajo Final Integrador se eligió el dominio *Mascota–Microchip*, inspirado en los sistemas de registro de animales y su identificación mediante microchips implantables.

La elección de este dominio permite representar de forma sencilla una relación uno a uno entre dos entidades (una mascota y su microchip), así como aplicar principios de programación orientada a objetos (herencia, encapsulamiento y polimorfismo) y acceso a datos mediante JDBC y el patrón DAO.

Además, el dominio facilita la implementación de transacciones ACID, simulando un caso real en el que el registro de una mascota y su microchip debe realizarse de manera conjunta para mantener la integridad de la base de datos. El desarrollo de este trabajo permitió la aplicación de conceptos de las materias Base de Datos y Programación II de la carrera, a través de un sistema simple pero funcional que podría evolucionar a un producto real (por ejemplo, un sistema de gestión para veterinarias o municipalidades).

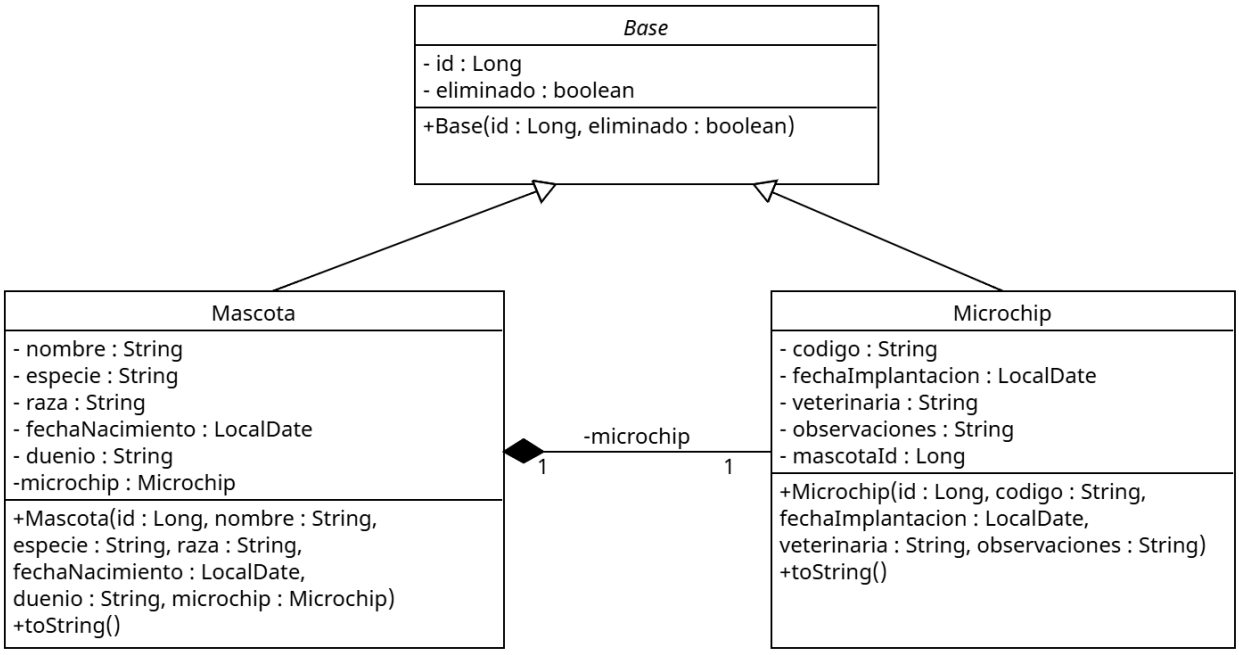
DISEÑO

Para respetar los lineamientos pedidos para desarrollar este trabajo practico se estableció una relación 1 a 1 entre las entidades Mascota y Microchip.

Cada mascota puede tener como máximo un microchip asignado y cada microchip pertenece exclusivamente a una mascota.

Esta decisión se basó tanto en el dominio real del problema (los microchips de identificación animal son únicos e intransferibles), como en la necesidad de mantener la integridad referencial y evitar duplicaciones en los registros.

DIAGRAMA UML



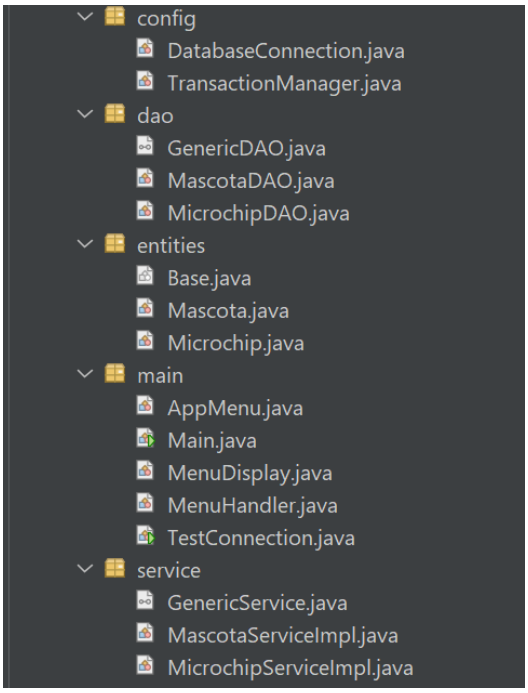
El siguiente diagrama UML representa la estructura principal del sistema, incluyendo las clases del dominio y sus relaciones:

- **Base**: clase abstracta que contiene los atributos comunes id y eliminado.
- **Mascota**: hereda de Base y modela los datos de una mascota (nombre, especie, raza, fechaNacimiento, duenio). Además, contiene una referencia a un objeto Microchip, lo que refleja la relación 1-1 del dominio.
- **Microchip**: también hereda de Base y almacena la información del dispositivo implantado (codigo, fechaImplantacion, veterinaria, observaciones). Posee el atributo mascotaId que actúa como **clave foránea única**, garantizando que cada microchip esté asociado a una única mascota.

En el diagrama se observa una asociación 1-1 entre Mascota y Microchip, donde cada instancia de Mascota puede tener un solo Microchip y viceversa. La composición (rombo negro) indica que el ciclo de vida del microchip está ligado a la mascota: si la mascota se elimina, también se elimina su microchip asociado.

ARQUITECTURA POR CAPAS

El sistema se desarrolló siguiendo el patrón de diseño DAO (multicapa) que permite separar claramente la responsabilidad de cada parte del programa. Esta decisión facilita el mantenimiento, la reutilización de código y la escalabilidad futura del sistema.



La aplicación está estructurada en 5 paquetes:

/**entitites**: contiene las clases que representan las entidades del sistema (es decir, los objetos del mundo real definidos en el dominio). Cada clase define sus atributos, constructores, getters, setters y métodos toString() propios.

/**service**: esta capa actúa como intermediaria entre la capa DAO y la interfaz del usuario. Contiene las clases MicrochipServiceImpl y MascotaServiceImpl que aplican las reglas de negocio y las validaciones necesarias antes de acceder a la base de datos: fechas futuras, id's válidos, etc. Esta capa también es la encargada de las gestionar las transacciones ACID mediante la clase TransactionManager.

/**dao**: es la capa encargada de interactuar con la base de datos MySQL. En este paquete se definen las clases MascotaDAO y MicrochipDAO. Estas clases contienen las operaciones CRUD utilizando JDBC y consultas SQL parametrizadas.

/**config**: Este paquete centraliza los parámetros de conexión. Contiene las clases encargadas de la conexión con la base de datos y la configuración general del sistema. La clase DatabaseConnection gestiona la conexión JDBC a db_mascotas utilizando el driver *com.mysql.cj.jdbc.Driver.*

/**main**: En esta capa se encuentra la lógica de interacción con usuario a través de la consola. Las clases MenuHandler y MenuDisplay permiten navegar los menús y ejecutar operaciones CRUD sobre mascotas y microchips. Esta capa se comunica exclusivamente con la capa service, sin acceder directamente a la base de datos, respetando así la independencia entre capas.

ESTRUCTURA DE LA BASE

El sistema utiliza persistencia mediante una base de datos relacional MySQL llamada **db_mascotas**, diseñada para almacenar de forma permanente la información de mascotas y sus respectivos microchips.

La base de datos esta compuesta por dos tablas principales: *mascotas* y *microchips*.

mascotas contiene los siguientes atributos: id (clave primaria), eliminado, nombre, especie, raza, fecha_nacimiento, duenio.

microchips contiene los atributos: id (clave primaria), eliminado, codigo, fecha_implantacion, veterinaria, observaciones, mascota_id (clave foránea).

La relación entre ambas tablas es **uno a uno (1→1)**, donde microchips.mascota_id actúa como **clave foránea única** que referencia a mascotas.id.
Se utiliza la restricción ON DELETE CASCADE y ON UPDATE CASCADE para mantener la integridad referencial: si una mascota es eliminada lógicamente, su microchip asociado también queda inactivo.

ORDEN DE OPERACIONES

El acceso a los datos se realiza siguiendo un flujo controlado por capas:

1. El usuario interactúa con el menú en consola (MenuHandler.java)
2. La acción seleccionada se envía al servicio que corresponda: MascotaServiceImpl o MicrochipServiceImpl.
3. El servicio valida los datos y llama al DAO para realizar las operaciones SQL.
4. Los resultados se devuelven a la capa de presentación para mostrar los mensajes al usuario.

TRANSACCIONES (commit y rollback)

La clase TransactionManager se encarga de manejar manualmente las transacciones (por ejemplo, insertarConMicrochip()):

1. Inicio de la transacción: primero desactiva el autocommit de la conexión mediante setAutoCommit(false).

2. Commit: si ambas inserciones (mascota y microchip) se ejecutan correctamente se confirma la transacción (commit()) garantizando que los datos se almacenen de forma permanente.
3. Rollback: si ocurre una excepción (por ejemplo, un código de microchip duplicado o un error de validación) se revierte toda la transacción mediante rollback asegurando que ninguna de las tablas quede con datos inconsistentes.
4. El TransactionManager cierra la conexión en un bloque *finally,* garantizando que los recursos sean liberados sin importar el resultado.

UTN
TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

VALIDACIONES Y REGLAS DE NEGOCIO

El sistema implementa validaciones con el objetivo de garantizar la coherencia e integridad de los datos antes de ejecutar cualquier operación.

Validaciones en la capa de servicio

- Campos obligatorios: no se permite dejar vacíos *nombre*, *especie*, *duenio* o *veterinaria*.
- Fechas válidas: las fechas de nacimiento e implantación no pueden ser futuras.
- Identificadores: se valida que los IDs sean positivos y no nulos antes de leer, actualizar o eliminar.
- Código único: no se pueden registrar microchips con un mismo código (restricción UNIQUE).
- Relación 1 a 1: se impide crear un nuevo microchip si la mascota ya tiene uno asignado.

Reglas de negocio aplicadas

- El borrado es lógico, no físico. Los registros se marcan con eliminado = true para mantener trazabilidad.
- Las operaciones combinadas (crear, actualizar o eliminar mascota + microchip) se ejecutan en transacciones ACID.
- Si ocurre un error durante una transacción, se realiza rollback para mantener la consistencia entre tablas.
- La lógica de negocio se encapsula en la capa service, separada del acceso a datos (dao) y de la interfaz (main).

PRUEBAS REALIZADAS

Se realizaron distintas pruebas para verificar la correcta funcionalidad del sistema, tanto a nivel de consola como en la base de datos MySQL.

```
------ MENU PRINCIPAL ------
[1] Gestionar MASCOTAS
[2] Gestionar MICROCHIPS
[3] Mostrar mascotas con microchips (vista)
[4] Transacciones
[0] Salir
Seleccione una opción:
```

1. Creación de mascota y microchip mediante opción 4 (Transacciones):

```
------ TRANSACCIONES ------
[1] Crear mascota y microchip (Transacción ACID)
[2] Actualizar mascota y microchip
[3] Eliminar mascota y microchip asociado
[0] Volver al menu principal
------ ---------- ------
Seleccione una opción:
```

Se ingresan los datos para mascota y microchip

```
Seleccione una opción: 1
---- Crear mascota y microchip [Transaccion ACID] ----
---- Ingrese los datos de la mascota ----
Nombre: Dakota
Especie: Perro
Raza: Cruza
Dueño: Tomas
Fecha de nacimiento (en formato AAAA-MM-DD o vacío): 2022-04-15
---------- CREAR MICROCHIP ----------
---- Ingrese los datos del microchip ----
Codigo: DKT567
Fecha de implantacion (en formato AAAA-MM-DD o vacío): 2022-08-22
Veterinaria: Mascotas al ataque
Observaciones: Se porto bien. Se agenda cita para control.
Iniciando transacción...
---- Mascota y microchip creados correctamente----
```

```
---- Datos de la Mascota creada ----
Mascota[ ID: null | Nombre: Dakota | Especie: Perro | Raza: Cruza | Fecha de nacimiento: 2022-04-15 | Dueño: Tomas | Eliminado: false ]

---- Datos del Microchip creado ----
Microchip[ ID: null | Codigo: DKT567 | Fecha de implantacion: 2022-08-22 | Veterinaria: Mascotas al ataque ]
```

Se confirma en DBeaver que la mascota y el microchip se hayan asociado correctamente (imagen dividida para legibilidad):

SELECT * from vista_mascotas_con_microd | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla

	id_mascota	nombre	especie	raza	fecha_nacim	duenio	i
1	1	Dakota	Perro	Cruza	2022-04-15	Tomas	

duenio	id_microchip	codigo	fecha_implantacion	veterinaria	observaciones
Tomas	1	DKT567	2022-08-22	Mascotas al ataque	Se porto bien. Se age

2. Se realiza prueba para intentar crear una mascota (mediate CRUD) y asociarle un microchip que ya existe (código **DKT567 ya pertenece a mascota Dakota**).

```
------------ CREAR MASCOTA ------------
---- Ingrese los datos de la mascota ----
Nombre: Miel
Especie: Gato
Raza: Persa
Dueño: Pablo
Fecha de nacimiento (en formato AAAA-MM-DD o vacío):
¿Desea registrar un microchip para esta mascota? (s/n): s
---- Ingrese los datos del microchip ----
Código: DKT567
Fecha de implantación (AAAA-MM-DD o vacío):
Veterinaria: Animal Care
Observaciones: Se implanto chip en lado izquierdo
```

Esta operación arroja un mensaje de error ya que la restricción de unicidad para el código impide asociar un microchip con código ya existente a otra mascota. En este caso, el error se muestra dentro del menú porque la excepción se captura en la capa de presentación (MenuHandler), que imprime el mensaje de error y luego redibuja el menú principal:


```
------ GESTION DE MASCOTAS ------
[1] Crear mascota
[2] Listar mascota
[3] Buscar mascota por ID
[4] Actualizar mascota
Error al registrar mascota con microchip: Error al insertar el microchip con código: DKT567
[5] Eliminar mascota
[6] Buscar mascota por nombre
[0] Volver al menu principal
------ ---------------------- ------
```

Prueba de consulta sobre la vista creada.

```
------ MENU PRINCIPAL ------
[1] Gestionar MASCOTAS
[2] Gestionar MICROCHIPS
[3] Mostrar mascotas con microchips (vista)
[4] Transacciones
[0] Salir
Seleccione una opción: 3
Mascota: Dakota (Perro) -- Dueño: Tomas -- Chip: DKT567 -- Veterinaria: Mascotas al ataque
```

Resultados de operaciones CRUD

Creación de mascotas adicionales y luego listarlas:

```
Seleccione una opción: 2
---- MASCOTAS ----
Mascota[ ID: 1 | Nombre: Dakota | Especie: Perro | Raza: Cruza | Fecha de nacimiento: 2022-04-15 | Dueño: Tomas | Eliminado: false ]
Mascota[ ID: 3 | Nombre: Akela | Especie: Perro | Raza: Dobberman | Fecha de nacimiento: 2019-05-22 | Dueño: Tomas | Eliminado: false ]
Mascota[ ID: 4 | Nombre: Garfield | Especie: Gato | Raza: Cruza | Fecha de nacimiento: 2024-05-03 | Dueño: Jon Arbuckle | Eliminado: false ]
```

Listar los microchips:

```
Seleccione una opción: 2
---- MICROCHIPS ----
1. Microchip[ ID: 1 | Codigo: DKT567 | Fecha de implantacion: 2022-08-22 | Veterinaria: Mascotas al ataque ]
2. Microchip[ ID: 3 | Codigo: AKL123 | Fecha de implantacion: 2023-08-21 | Veterinaria: Veterinaria El Grego ]
3. Microchip[ ID: 4 | Codigo: GFD888 | Fecha de implantacion: null | Veterinaria: Peluditos ]
```

Eliminar una mascota (con ID 4) – mediante borrado lógico.

```
---- Eliminar mascota ----
Ingrese el ID de la mascota a eliminar (mediante baja logica): 4
Mascota encontrada: Mascota[ ID: 4 | Nombre: Garfield | Especie: Gato |
Raza: Cruza | Fecha de nacimiento: 2024-05-03 | Dueño: Jon Arbuckle | Eliminado: false ]

¿Esta seguro de que desea eliminar a la mascota Garfield?
Ingrese s/n: s
Eliminando mascota...
Mascota con ID: 4 eliminada correctamente (baja lógica).
```

Actualizar datos de una mascota (con ID 3)

```
Seleccione una opción: 4
---- Actualizar mascota ----
Ingrese el ID de la mascota a actualizar: 3
Mascota encontrada. Ingrese los datos de la mascota:
(Dejar entrada vacia (enter) para conservar valor actual)
Nuevo nombre [actual: Akela]: Panda
Nueva especie [actual: Perro]: Gato
Nueva raza [actual: Dobberman]: Cruza
Nuevo dueño [actual: Tomas]:
Nueva fecha de nacimiento [actual: 2019-05-22]: 2022-11-09
Actualizando datos...
La mascota con ID 3 ha sido actualizada correctamente.

---- MASCOTAS ----
Mascota[ ID: 1 | Nombre: Dakota | Especie: Perro | Raza: Cruza | Fecha de nacimiento: 2022-04-15 | Dueño: Tomas | Eliminado: false ]
Mascota[ ID: 3 | Nombre: Panda | Especie: Gato | Raza: Cruza | Fecha de nacimiento: 2022-11-09 | Dueño: Tomas | Eliminado: false ]
```

Se realizaron pruebas exhaustivas para verificar que todo el sistema funcione correctamente y como se espera. Por la limitación en la extensión de este informe no se incluyen todas las pruebas, solo las más importantes. Durante el

desarrollo del video de este trabajo practico se exponen otras pruebas realizadas.

CONCLUSIONES Y MEJORAS FUTURAS

El desarrollo del sistema *Mascota–Microchip* permitió aplicar los principales conceptos de la programación orientada a objetos, acceso a bases de datos con JDBC, y manejo transaccional.
La estructura en capas demostró ser eficaz para mantener la separación de responsabilidades y facilitar las pruebas.

SUGERENCIAS DE MEJORA

1. Incorporar un módulo sobre dueños o veterinarias con relación 1 a muchos.
2. Implementar una interfaz gráfica (mediante Swing o JavaFX).
3. Una mejora importante para versiones futuras del sistema sería normalizar aún más la base de datos e incorporar nuevas tablas relacionadas (duenios o veterinarias). Una estructura más normalizada y modular permitiría al sistema soportar grandes volúmenes de información, optimizar las consultas mediante índices, y prepararlo para integraciones futuras (por ejemplo, reportes estadísticos, paneles administrativos o conexión con servicios web externos).
4. Expandir el sistema para que realice informes estadísticos y/o exporte datos en formato de Excel/ CSV.