UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Final Integrador

Sistema de Gestión Veterinaria

Alumnos

Jennifer Franco (jennyfranco31.jf@gmail.com)

Jonathan Franco (nahuelfranco7@icloud.com)

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Arquitectura y Sistemas Operativos

Docente Titular

Alberto Córtez

Docente Tutor

Neyén Bianchi

20 de Noviembre de 2025

UNIVERSIDAD TECNOLOGICA NACIONAL

1. Integrantes del Equipo **3**
2. Elección del Dominio y Justificación **3**
3. Diseño y Modelado **3**
3.1. Decisiones de Diseño Clave 3
3.2. Diagrama UML de Clases 4
4. Arquitectura del Sistema **4**
4.1. Paquetes y Responsabilidades 4
5. Persistencia y Transacciones **5**
5.1. Estructura de la Base de Datos 5
5.2. Gestión de Transacciones 5
6. Validaciones y Reglas de Negocio **5**
7. Pruebas Realizadas **6**
7.1. Caso de Uso: Transacción Exitosa 6
7.2. Caso de Uso: Rollback (Prueba de Fallo) 7
7.3. Consultas SQL de Verificación 7
8. Conclusiones y Mejoras Futuras **8**
9. Referencias y Herramientas **8**
10. Video Demostrativo **8**

UNIVERSIDAD TECNOLOGICA NACIONAL

1. Integrantes del Equipo

- **Franco, Jennifer** (Legajo: [Completar]) Datos]
- **Franco, Jonathan**

2. Elección del Dominio y Justificación

Para el desarrollo de este Trabajo Final Integrador, el equipo ha seleccionado el dominio: **Mascota → Microchip**.

Adicionalmente, para dotar al sistema de mayor realismo y coherencia funcional, se decidió implementar una entidad superior, **Dueño**, estableciendo una jerarquía completa de datos.

Justificación de la elección: Este dominio fue seleccionado porque representa un escenario de la vida real altamente aplicable en clínicas veterinarias o sistemas de control municipal. Permite modelar claramente las relaciones requeridas por la cátedra:

1. **Relación 1 a 1 (Unidireccional):** Una Mascota posee un único Microchip de identificación, y el Microchip sirve para identificar a esa única mascota. Es un ejemplo canónico de relación 1-a-1 estricta.
2. **Relación 1 a Muchos:** Un Dueño puede tener múltiples mascotas, lo que añade complejidad y riqueza a las consultas SQL (JOINs) y validaciones de integridad.

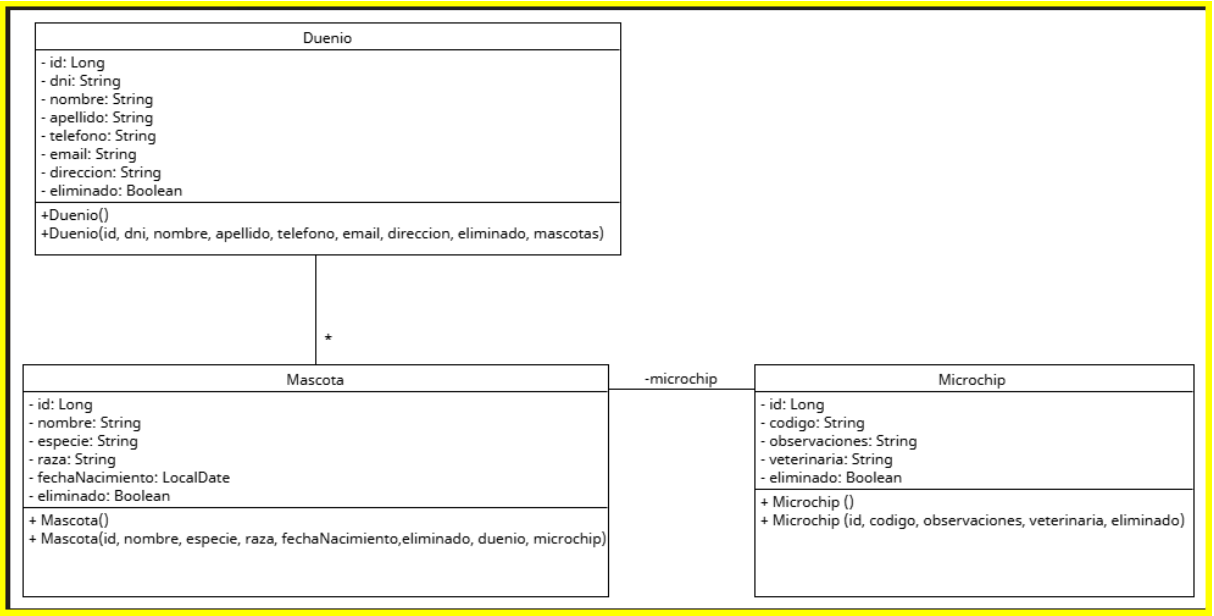
3. Diseño y Modelado

3.1. Decisiones de Diseño Clave

Para la persistencia de la relación 1-a-1 entre Mascota y Microchip, se evaluaron las opciones propuestas por la cátedra y se optó por la estrategia de Clave Foránea Única en la Tabla B (Microchip).

- Implementación: La tabla microchips contiene una columna mascota_id con una restricción UNIQUE.
- **Justificación:** Esta decisión desacopla la creación de la mascota de la del chip a nivel de estructura, pero garantiza la unicidad a nivel de base de datos. Es la opción recomendada por la cátedra por su robustez.

3.2. Diagrama UML de Clases



Se destaca el uso de **Encapsulamiento** (atributos privados, getters/setters públicos) y la implementación de la **Relación Unidireccional** (Mascota tiene una referencia a Microchip, pero Microchip no conoce a su Mascota).

4. Arquitectura del Sistema

El proyecto se ha estructurado siguiendo una **Arquitectura en Capas** estricta para garantizar la separación de responsabilidades y la mantenibilidad del código .

4.1. Paquetes y Responsabilidades

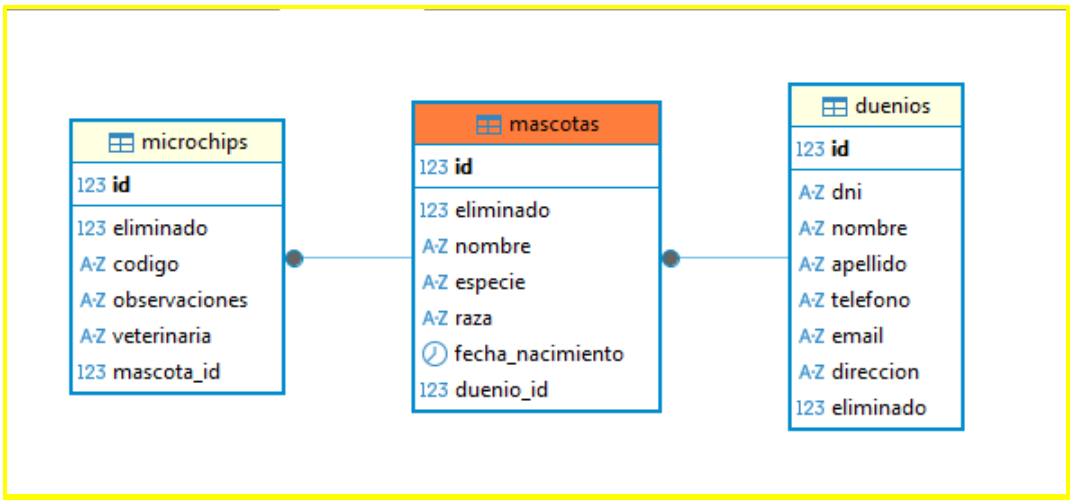
1. **config:** Contiene la clase DatabaseConnectionPool. Se implementó el patrón Singleton mediante la librería HikariCP para gestionar un pool de conexiones eficiente, mejorando el rendimiento frente a la apertura/cierre manual de conexiones JDBC.
2. **entities** (Modelo): Contiene los POJOs (Duenio, Mascota, Microchip). Son clases anémicas que solo transportan datos.
3. **dao** (Acceso a Datos):
    - Contiene la interfaz genérica GenericDAO<T> y las interfaces específicas (DuenioDAO, etc.).
    - Contiene las implementaciones (...DaoImpl). Es la única capa que contiene código SQL y dependencias de JDBC (PreparedStatement, ResultSet).
4. **service** (Lógica de Negocio):
    - Contiene la lógica "inteligente". Valida reglas de negocio, orquesta múltiples DAOs y, fundamentalmente, gestiona las transacciones.
5. **main** (Presentación):
    - Contiene el punto de entrada (Main), el controlador de flujo (MenuHandler) y la vista (MenuDisplay). Se encarga exclusivamente de la interacción con el usuario vía consola.

UNIVERSIDAD TECNOLOGICA NACIONAL

5. Persistencia y Transacciones

5.1. Estructura de la Base de Datos

Se utilizó MySQL como motor de base de datos. Las tablas cuentan con restricciones de integridad (FOREIGN KEY, UNIQUE) y un campo eliminado (BOOLEAN) para soportar la Baja Lógica.



5.2. Gestión de Transacciones

El requisito más crítico del TFI, la transacción 1-a-1, se implementó en la clase MascotaServiceImpl.

Flujo de la operación **crearMascotaCompleta**:

1. El Service solicita una **conexión al pool** y deshabilita el auto-commit (conn.setAutoCommit(false)).
2. **Paso 1:** Llama a MascotaDao.crear() pasando la conexión. Se inserta la mascota y se recupera su ID generado.
3. **Paso 2:** Llama a MicrochipDao.crear() pasando la conexión y el ID de la mascota recién creada.
4. **Cierre:** Si ambos pasos son exitosos, se ejecuta conn.commit().
5. **Manejo de Errores:** Si ocurre cualquier error (ej. código de chip duplicado), se captura la excepción y se ejecuta conn.rollback(), garantizando que no se guarde una mascota sin chip o viceversa.

6. Validaciones y Reglas de Negocio

Toda la lógica de validación se centralizó en la capa Service para no contaminar los DAOs ni la UI.

Reglas Implementadas:

1. Integridad Referencial Lógica: No se permite eliminar un Duenio si este posee mascotas activas (eliminado = false). El servicio verifica esto mediante mascotaDao.contarMascotasActivasPorDuenio.

2. **Unicidad:** Se valida que el DNI del dueño y el Código del Microchip sean únicos en el sistema antes de intentar la inserción.
3. **Campos Obligatorios:** Se valida que campos críticos (Nombre, DNI, Código Chip) no sean nulos ni vacíos.
4. **Baja en Cascada:** Al eliminar una Mascota (lógicamente), el sistema elimina automáticamente su Microchip asociado dentro de la misma transacción.

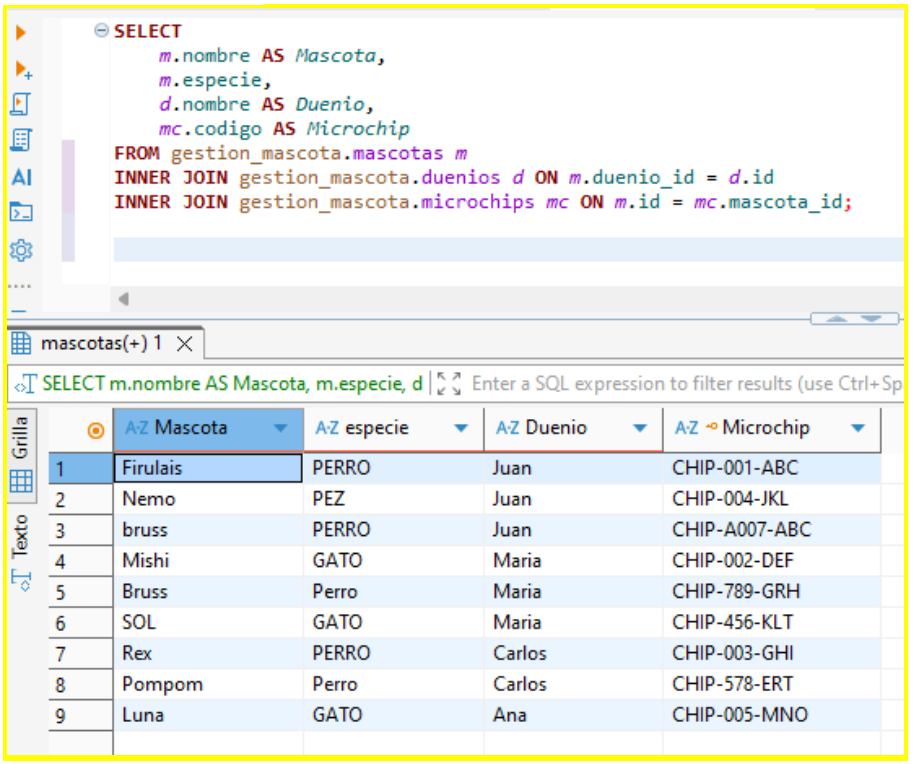
7. Pruebas Realizadas

Se realizaron pruebas exhaustivas del flujo CRUD mediante la interfaz de consola.

7.1. Caso de Uso: Transacción Exitosa

```
--- Gestión de Mascotas ---
 8. Registrar Mascota y Microchip
 9. Listar Todas las Mascotas
 10. Listar Mascotas por Dueño
 11. Eliminar Mascota (Baja en Cascada)
------------------------------------------------------------
 0. Salir del Sistema
====================================
>> Ingrese una opción: 8

--- 8. Registrar Mascota (con Microchip) ---
Ingrese el ID del Dueño de la mascota: 3
Dueño encontrado: Carlos Lopez
Nombre de la mascota: Pompom
Especie: Perro
Raza (opcional): Caniche
Ingrese Fecha de Nacimiento (AAAA-MM-DD) [Opcional]: 2019-05-31
Ingrese Código del Microchip (obligatorio): CHIP-578-ERT
Ingrese Veterinaria (opcional): Veterinaria Sur

--- ¡ÉXITO! ---
Mascota creada con ID: 9 y Microchip ID: 9
----------------
```

	id	eliminado	nombre	especie	raza	fecha_nacimiento	duenio_id
1	1	0	Firulais	PERRO	Labrador	2020-05-20	1
2	2	0	Mishi	GATO	Siames	2019-10-15	2
3	3	0	Rex	PERRO	Ovejero	2021-01-01	3
4	4	0	Nemo	PEZ	Dorado	2023-03-10	1
5	5	0	Luna	GATO	Callejero	2022-07-07	4
6	6	0	Bruss	Perro	Callejero	2018-07-01	2
7	7	0	SOL	GATO	[NULL]	[NULL]	2
8	8	1	bruss	PERRO	callejerO	[NULL]	1
9	9	0	Pompom	Perro	Caniche	2019-05-31	3



```sql
SELECT
    m.nombre AS Mascota,
    m.especie,
    d.nombre AS Duenio,
    mc.codigo AS Microchip
FROM gestion_mascota.mascotas m
INNER JOIN gestion_mascota.duenios d ON m.duenio_id = d.id
INNER JOIN gestion_mascota.microchips mc ON m.id = mc.mascota_id;
```

mascotas(+) 1

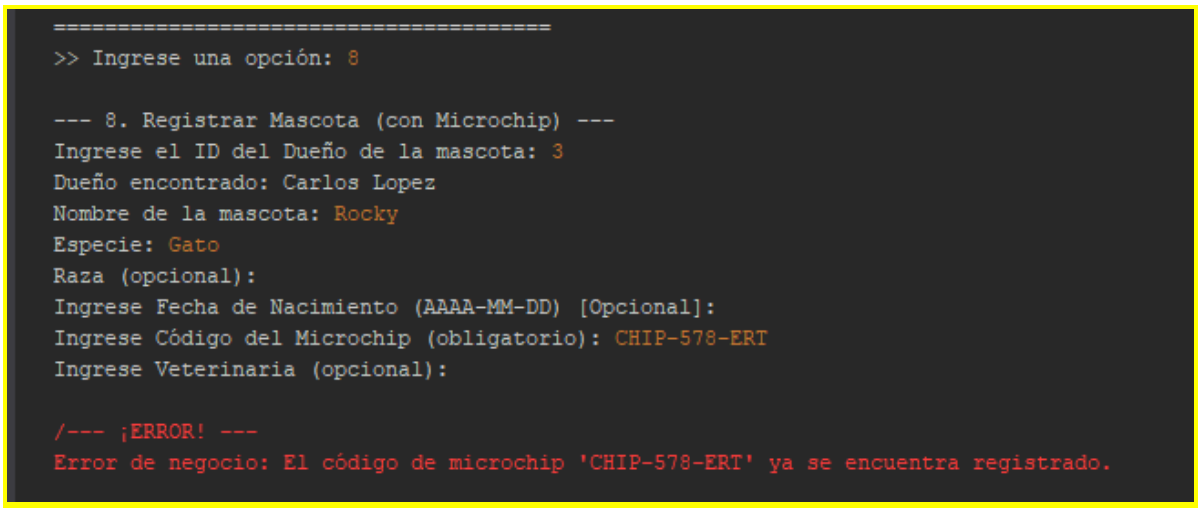
SELECT m.nombre AS Mascota, m.especie, d | Enter a SQL expression to filter results (use Ctrl+Sp

Grilla | Texto

	Mascota	especie	Duenio	Microchip
1	Firulais	PERRO	Juan	CHIP-001-ABC
2	Nemo	PEZ	Juan	CHIP-004-JKL
3	bruss	PERRO	Juan	CHIP-A007-ABC
4	Mishi	GATO	Maria	CHIP-002-DEF
5	Bruss	Perro	Maria	CHIP-789-GRH
6	SOL	GATO	Maria	CHIP-456-KLT
7	Rex	PERRO	Carlos	CHIP-003-GHI
8	Pompom	Perro	Carlos	CHIP-578-ERT
9	Luna	GATO	Ana	CHIP-005-MNO

7.2. Caso de Uso: Rollback (Prueba de Fallo)

Se intentó registrar una mascota con un código de microchip ya existente.

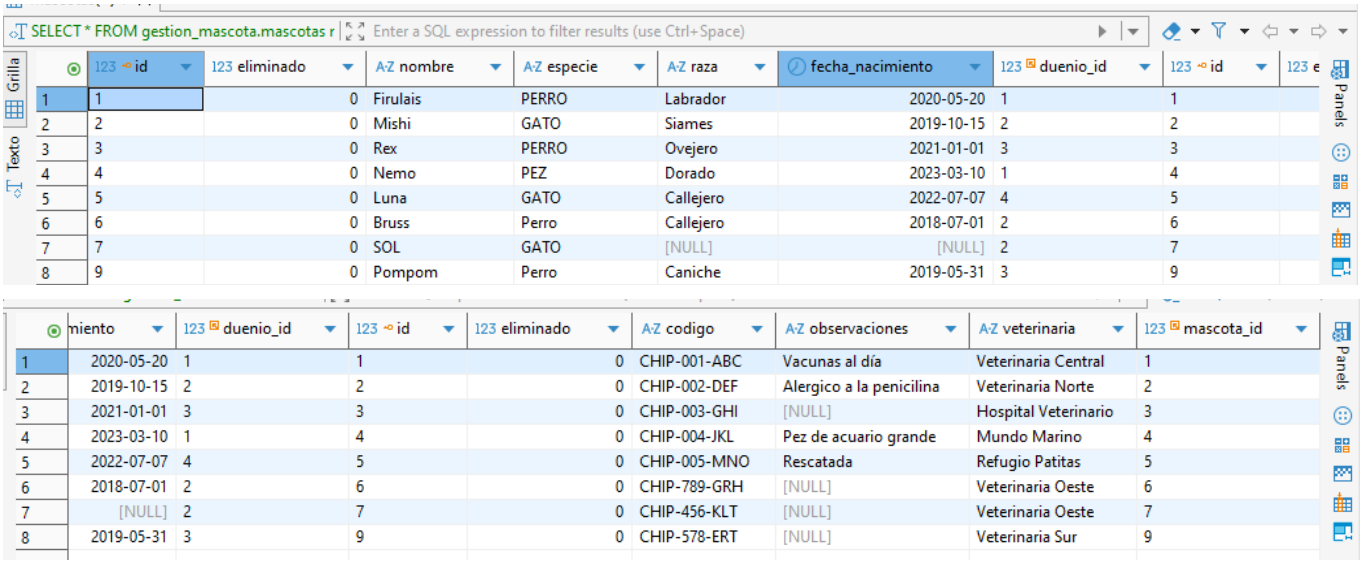


```
========================================
>> Ingrese una opción: 8

--- 8. Registrar Mascota (con Microchip) ---
Ingrese el ID del Dueño de la mascota: 3
Dueño encontrado: Carlos Lopez
Nombre de la mascota: Rocky
Especie: Gato
Raza (opcional):
Ingrese Fecha de Nacimiento (AAAA-MM-DD) [Opcional]:
Ingrese Código del Microchip (obligatorio): CHIP-578-ERT
Ingrese Veterinaria (opcional):

/--- ¡ERROR! ---
Error de negocio: El código de microchip 'CHIP-578-ERT' ya se encuentra registrado.
```

7.3. Consultas SQL de Verificación

Para validar la persistencia, se utilizaron consultas directas en DBeaver:

```sql
SELECT * FROM gestion_mascota.mascotas m
JOIN gestion_mascota.microchips mc ON m.id = mc.mascota_id
WHERE m.eliminado = 0;
```



SELECT * FROM gestion_mascota.mascotas r — Enter a SQL expression to filter results (use Ctrl+Space)

	id	eliminado	nombre	especie	raza	fecha_nacimiento	duenio_id	id	e
1	1	0	Firulais	PERRO	Labrador	2020-05-20	1	1	
2	2	0	Mishi	GATO	Siames	2019-10-15	2	2	
3	3	0	Rex	PERRO	Ovejero	2021-01-01	3	3	
4	4	0	Nemo	PEZ	Dorado	2023-03-10	1	4	
5	5	0	Luna	GATO	Callejero	2022-07-07	4	5	
6	6	0	Bruss	Perro	Callejero	2018-07-01	2	6	
7	7	0	SOL	GATO	[NULL]	[NULL]	2	7	
8	9	0	Pompom	Perro	Caniche	2019-05-31	3	9	

	miento	duenio_id	id	eliminado	codigo	observaciones	veterinaria	mascota_id
1	2020-05-20	1	1	0	CHIP-001-ABC	Vacunas al día	Veterinaria Central	1
2	2019-10-15	2	2	0	CHIP-002-DEF	Alergico a la penicilina	Veterinaria Norte	2
3	2021-01-01	3	3	0	CHIP-003-GHI	[NULL]	Hospital Veterinario	3
4	2023-03-10	1	4	0	CHIP-004-JKL	Pez de acuario grande	Mundo Marino	4
5	2022-07-07	4	5	0	CHIP-005-MNO	Rescatada	Refugio Patitas	5
6	2018-07-01	2	6	0	CHIP-789-GRH	[NULL]	Veterinaria Oeste	6
7	[NULL]	2	7	0	CHIP-456-KLT	[NULL]	Veterinaria Oeste	7
8	2019-05-31	3	9	0	CHIP-578-ERT	[NULL]	Veterinaria Sur	9

8. Conclusiones y Mejoras Futuras

El desarrollo de este TFI permitió consolidar el conocimiento sobre JDBC y la importancia de la arquitectura en capas. El mayor desafío fue la correcta implementación de la transacción manual sin frameworks (como Hibernate), lo cual brindó un entendimiento profundo del ciclo de vida de una conexión y el manejo de errores SQL.

Mejoras Futuras:

1. Implementar una interfaz gráfica (GUI) utilizando JavaFX o Swing.
2. Agregar un sistema de reportes (ej. exportar listado a PDF/Excel).
3. Migrar la persistencia a JPA/Hibernate para simplificar el mapeo objeto-relacional.

9. Referencias y Herramientas

- **Lenguaje:** Java JDK 21.
- **Base de Datos:** MySQL Community Server 8.0.
- **Librerías:**
  - mysql-connector-j-8.4.0.jar (Driver JDBC).
  - HikariCP (Connection Pool).
  - SLF4J (Logging facade para HikariCP).
- **IDE:** Apache NetBeans.
- **Herramientas de Apoyo:** DBeaver (Gestión de BD), Google Gemini (Asistencia en refactorización de código y documentación).

10. Video Demostrativo

Link del video en youtube: https://youtu.be/TCYOXfE_YZk