Trabajo Final Integrador (TFI) - Programación II

Alumnos - Comisión 10

Berrone Lanza Lina Lucia
Bayurk Mara Valentina
Cabrera Dario Ezequiel

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.
Programación II

Coordinador
Carlos Martinez

Docente Tutor
Francisco Quarñolo

15 de Noviembre de 2025

Índice

1. Elección del dominio y justificación

El dominio elegido es “Gestión de mascotas y microchips en una clínica veterinaria”.

- Es un dominio con entidades claras: una mascota puede o no tener microchip; cada microchip tiene un código único; la clínica necesita llevar control de implantaciones y propietarios.
- Permite practicar conceptos clave: modelado entidad-relación, decisiones sobre cardinalidades (1→1), manejo de integridad referencial, transaccionalidad (inserción coordinada de mascota + microchip), validaciones y reglas de negocio.
- Es un dominio pequeño pero con casos reales (alta, baja lógica, actualización, consultas relacionales).

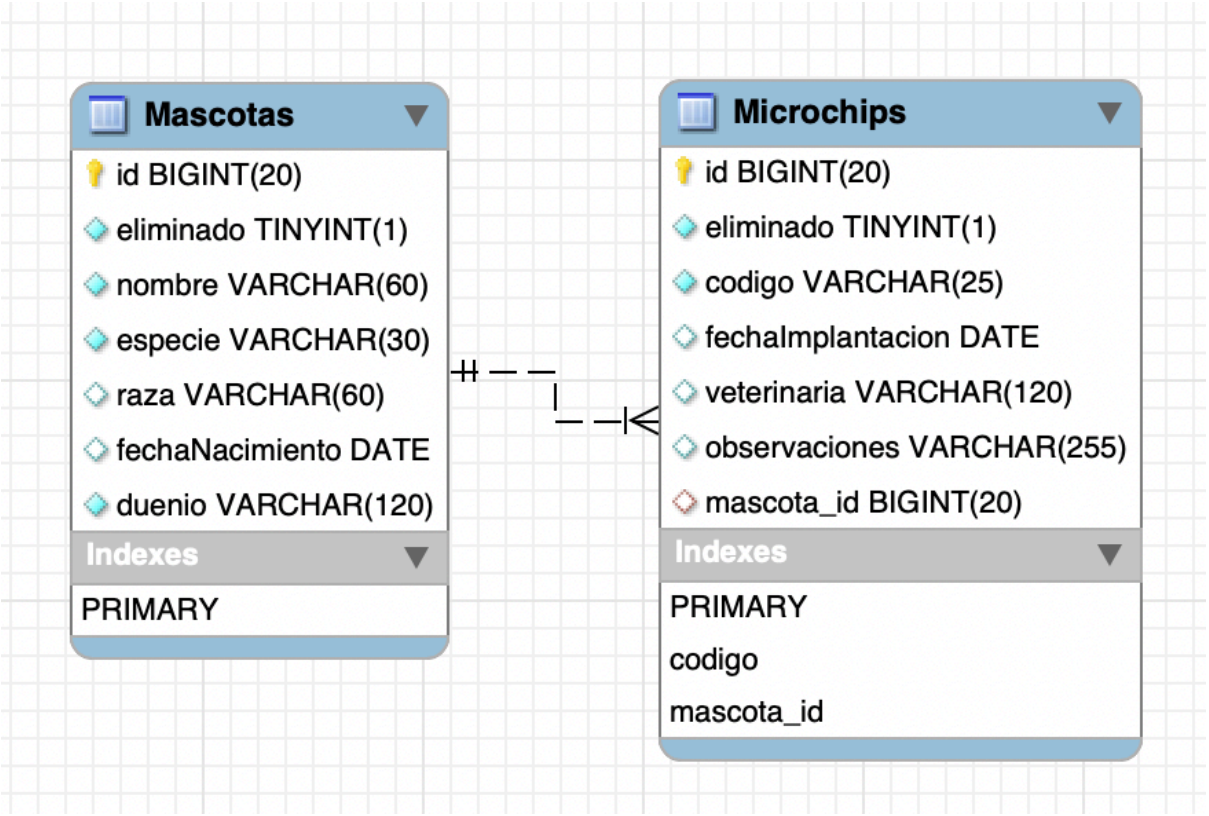
Las entidades principales son:

1. Mascotas (Clase A) → representa a la mascota registrada.
2. Microchips (Clase B) → representa a un identificador permanente implantado.

Se modeló la relación como uno a uno (1→1) con la siguiente implementación práctica:

- Tabla Mascotas con su propia PK (id autoincremental).
- Tabla Microchips con su propia PK (id autoincremental) y una columna mascota_id marcada como UNIQUE y FK a Mascotas(id).

UML



2. Arquitectura por capas

El proyecto se organizó en paquetes siguiendo el patrón DAO + Service:

- config/
    - DatabaseConnection.java: centraliza la configuración de conexión JDBC a MySQL.
    - Permite modificar credenciales o URL sin afectar el resto del código.
- models/
    - Mascota.java y Microchip.java: representan las entidades con atributos equivalentes a las columnas SQL.
    - Incluyen constructores, getters/setters y toString() para manipulación en memoria.
- dao/
    - GenericDAO<T>: define operaciones CRUD genéricas (crear, leer, leerTodos, actualizar, eliminar).
    - DAOs concretos (MascotaDAO, MicrochipDAO) implementan estas operaciones con PreparedStatement.
- service/
    - MascotasServiceImpl y MicrochipsServiceImpl: orquestan transacciones, validan reglas de negocio y coordinan operaciones entre DAOs.
    - Implementan commit/rollback para garantizar atomicidad.
- main/
    - MenuHandlers y Main: ofrecen un menú de consola para CRUD completo, búsquedas y manejo de errores.

3. Persistencia y transacciones

La persistencia se implementó en MySQL mediante scripts SQL que crean las tablas y cargan datos de prueba. Las decisiones clave fueron:

- Tabla Mascotas (Clase A):
    - PK: id autoincremental.
    - Baja lógica: campo eliminado BOOLEAN DEFAULT FALSE.
    - Campos obligatorios: nombre, especie, duenio.
- Tabla Microchips (Clase B):
    - PK: id autoincremental.
    - Baja lógica: campo eliminado BOOLEAN DEFAULT FALSE.
    - código único para garantizar trazabilidad.
    - FK: mascota_id UNIQUE → asegura la relación 1→1.
    - ON DELETE CASCADE → mantiene integridad referencial.

Orden de operaciones en transacciones:

1. Abrir conexión con DatabaseConnection.getConnection().
2. Desactivar autocommit (conn.setAutoCommit(false)).
3. Ejecutar operaciones compuestas:
    - Crear mascota (mascotaDAO.crear(conn, mascota)).
    - Crear microchip asociado (microchipDAO.crear(conn, microchip, mascotaId)).
4. Confirmar con conn.commit() si todo es correcto.
5. Revertir con conn.rollback() si ocurre un error.
6. Restablecer autocommit y cerrar recursos.

Ejemplo en Java:

java

```
try (Connection conn = DatabaseConnection.getConnection()) {
   conn.setAutoCommit(false);


   Long mascotaId = mascotaDAO.crear(conn, mascota);
   microchip.setMascotaId(mascotaId);
   microchipDAO.crear(conn, microchip);


   conn.commit();
   System.out.println("Mascota y microchip insertados correctamente.");
} catch (Exception e) {
   conn.rollback();
   System.err.println("Error en la transacción: " + e.getMessage());
} finally {
   conn.setAutoCommit(true);
}
```

Esto asegura atomicidad: o se insertan ambos registros, o ninguno.

4. Validaciones y reglas de negocio

Las validaciones se implementaron en la capa Service:

- Campos obligatorios:
    - Mascota: nombre, especie, dueño.
    - Microchip: código único.
- Regla 1→1:
    - Se controla que cada mascota tenga como máximo un microchip.
    - El campo mascota_id en Microchips está marcado como UNIQUE.
- Baja lógica:
    - Se utiliza el campo eliminado en lugar de borrar físicamente los registros.
    - Esto permite mantener historial y evitar pérdida de datos.
- Manejo de excepciones:
    - Los DAO lanzan SQLException ante errores de base.
    - Los Service capturan y traducen los errores en mensajes claros para el menú.

5. Pruebas realizadas

Pruebas en el menú de consola

Se realizaron pruebas de las operaciones CRUD y de las reglas de negocio. Algunos ejemplos de salida en consola:

Alta de mascota con microchip:

```
========================================
<<< Iniciando Sistema de Veterinaria TFI >>>
========================================

--- SISTEMA DE VETERINARIA TFI ---
1. Registrar Mascota y Microchip
2. Listar todas las Mascotas
3. Buscar Mascota por ID (con Microchip)
4. Actualizar Mascota y Microchip
5. Eliminar Mascota
6. Listar todos los Microchips
0. Salir
-----------------------------------
Seleccione una opcion: 1

--- CREAR MASCOTA Y ASOCIAR MICROCHIP ---
Nombre de la Mascota: Cookie
Especie: Gato
Raza: Persa
Fecha Nacimiento (AAAA-MM-DD): 2022-12-03
Duenio: Dario Cabrera

--- DATOS DEL MICROCHIP (1:1) ---
Codigo UNICO del Microchip: UTPJ5487
Fecha Implantacion (AAAA-MM-DD): 2023-12-03
Veterinaria: Como Perro Y Gato
Observaciones: Ninguna

 EXITO: Mascota y Microchip creados en una sola transaccion.
   Mascota ID Generado: 6
```

Listado de mascotas activas:

```
--- SISTEMA DE VETERINARIA TFI ---
1. Registrar Mascota y Microchip
2. Listar todas las Mascotas
3. Buscar Mascota por ID (con Microchip)
4. Actualizar Mascota y Microchip
5. Eliminar Mascota
6. Listar todos los Microchips
0. Salir
-----------------------------------
Seleccione una opcion: 2

--- LISTADO DE MASCOTAS ACTIVAS ---
Mascota{id=1, nombre='Pili', especie='Gato', microchip=ABL123, eliminado=false}
Mascota{id=2, nombre='Piliguerta', especie='Gato', microchip=AGHP1456, eliminado=false}
Mascota{id=3, nombre='Nani', especie='Gato', microchip=GHT125, eliminado=false}
Mascota{id=5, nombre='Sombra', especie='Gato', microchip=ENTP456, eliminado=false}
Mascota{id=6, nombre='Cookie', especie='Gato', microchip=UTPJ5487, eliminado=false}
```

Actualización de una mascota:

```
--- SISTEMA DE VETERINARIA TFI ---
1. Registrar Mascota y Microchip
2. Listar todas las Mascotas
3. Buscar Mascota por ID (con Microchip)
4. Actualizar Mascota y Microchip
5. Eliminar Mascota
6. Listar todos los Microchips
0. Salir
-----------------------------------
Seleccione una opcion: 4
Ingrese el ID de la mascota a ACTUALIZAR: 5

--- Actualizando Mascota ID: 5 ---
Valor actual del Nombre: Sombra
Nuevo Nombre (Dejar vacio para mantener): Negra

--- Actualizando Microchip Codigo: ENTP456 ---
Nueva Veterinaria (Dejar vacio para mantener):
Mascota ID 5 actualizada con exito.
```

```
Mascota{id=5, nombre='Negra', especie='Gato', microchip=ENTP456, eliminado=false}
Mascota{id=6, nombre='Cookie', especie='Gato', microchip=UTPJ5487, eliminado=false}
```

Baja lógica de una mascota:

```
--- LISTADO DE MASCOTAS ACTIVAS ---
Mascota{id=1, nombre='Pili', especie='Gato', microchip=ABL123, eliminado=false}
Mascota{id=2, nombre='Piliguerta', especie='Gato', microchip=AGHP1456, eliminado=false}
Mascota{id=3, nombre='Nani', especie='Gato', microchip=GHT125, eliminado=false}
Mascota{id=5, nombre='Negra', especie='Gato', microchip=ENTP456, eliminado=false}
Mascota{id=6, nombre='Cookie', especie='Gato', microchip=UTPJ5487, eliminado=false}

--- SISTEMA DE VETERINARIA TFI ---
1. Registrar Mascota y Microchip
2. Listar todas las Mascotas
3. Buscar Mascota por ID (con Microchip)
4. Actualizar Mascota y Microchip
5. Eliminar Mascota
6. Listar todos los Microchips
0. Salir
-----------------------------------
Seleccione una opcion: 5
Ingrese el ID de la mascota para BAJA LOGICA: 1
Mascota ID 1 marcada como eliminada (Baja Logica).
```

Listado de Microchips:

```
--- SISTEMA DE VETERINARIA TFI ---
1. Registrar Mascota y Microchip
2. Listar todas las Mascotas
3. Buscar Mascota por ID (con Microchip)
4. Actualizar Mascota y Microchip
5. Eliminar Mascota
6. Listar todos los Microchips
0. Salir
-----------------------------------
Seleccione una opcion: 6

--- LISTADO DE MICROCHIPS ---
Microchip{id=1, codigo='ABL123', fechaImplantacion=2001-12-03, veterinaria='Como Perro Y gato', observaciones='Ninguna', mascotaId=1, eliminado=false}
Microchip{id=2, codigo='AGHP1456', fechaImplantacion=1993-12-03, veterinaria='Siempre felices', observaciones='Ninguna', mascotaId=2, eliminado=false}
Microchip{id=3, codigo='GHT125', fechaImplantacion=2022-12-03, veterinaria='Siempre Felices', observaciones='Ninguna', mascotaId=3, eliminado=false}
Microchip{id=4, codigo='HWOP1256', fechaImplantacion=2002-12-03, veterinaria='Siempre Feliz', observaciones='Ninguna', mascotaId=4, eliminado=false}
Microchip{id=5, codigo='ENTP456', fechaImplantacion=2023-12-03, veterinaria='Como Perro Y Gato', observaciones='Ninguna', mascotaId=5, eliminado=false}
Microchip{id=6, codigo='UTPJ5487', fechaImplantacion=2023-12-03, veterinaria='Como Perro Y Gato', observaciones='Ninguna', mascotaId=6, eliminado=false}
```

```
45 •  SELECT * FROM Mascotas;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	eliminado	nombre	especie	raza	fechaNacimiento	duenio
1	1	Pili	Gato	Gato	2000-12-03	Lina
2	0	Piliguerta	Gato	Calle	1992-12-03	Lina
3	0	Nani	Gato	Calle	2000-12-03	Mara
4	1	Maria	Gato	Calle	2000-12-03	Dari
5	0	Negra	Gato	Persa	2022-12-03	Lina
6	0	Cookie	Gato	Persa	2022-12-03	Dario Cabrera
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Esto demuestra que el campo eliminado funciona correctamente y que los registros no se borran físicamente.

6. Conclusiones y mejoras futuras

Conclusiones

- Se logró implementar correctamente la relación 1⟶1 unidireccional entre Mascota (A) y Microchip (B), tanto en el modelo SQL como en las clases Java.
- La arquitectura por capas (config, models, dao, service, main) permitió mantener un código limpio, organizado y escalable, cumpliendo con el patrón DAO + Service.
- La persistencia en MySQL con constraints (UNIQUE, FOREIGN KEY, ON DELETE CASCADE) garantizó la integridad referencial y evitó inconsistencias en los datos.
- El uso de transacciones con commit/rollback en la capa Service aseguró la atomicidad de las operaciones compuestas (ejemplo: creación de mascota + microchip).
- Se implementaron validaciones y reglas de negocio: campos obligatorios, unicidad de microchip, baja lógica mediante el campo eliminado.
- Las pruebas realizadas demostraron que el menú de consola es funcional, robusto ante entradas inválidas y capaz de ejecutar CRUD completo, búsquedas y reportes.

En síntesis, el proyecto cumple con los criterios de evaluación: correctitud funcional, diseño por capas, calidad de código, persistencia con integridad, documentación clara y pruebas verificables.

Mejoras futuras

Aunque el sistema cumple con los requisitos actuales, se identifican posibles mejoras:

1. Validaciones adicionales
    - Verificar formatos de campos (ejemplo: email del dueño, fecha de nacimiento).
    - Implementar reglas de negocio más complejas (ejemplo: edad mínima de la mascota para implantar microchip).
2. Interfaz gráfica
    - Migrar el menú de consola a una interfaz gráfica (Swing, JavaFX o web con Spring Boot).
    - Esto mejoraría la experiencia del usuario final.
3. Extensión del dominio
    - Incorporar nuevas entidades relacionadas, como Veterinario → Turno o Propietario → Dirección.
    - Esto permitiría ampliar el sistema hacia una gestión clínica más completa.

4. Seguridad avanzada
    - Implementar autenticación de usuarios para el acceso al sistema.
    - Registrar auditorías de operaciones (quién creó, modificó o eliminó registros).
5. Persistencia mejorada
    - Uso de índices en campos de búsqueda frecuentes (ejemplo: duenio, codigo).
    - Optimización de consultas para grandes volúmenes de datos.

7. Fuentes y herramientas utilizadas

Para el desarrollo del Trabajo Final Integrador se utilizaron las siguientes herramientas y recursos:

- Lenguaje y entorno de desarrollo
    - Java 21 (recomendado por la consigna).
    - IDE: NetBeans (para compilación, ejecución y depuración).
- Base de datos
    - MySQL 8.0.
    - Scripts SQL para creación de tablas (Mascotas, Microchips) y carga de datos masivos.
    - Cliente MySQL Workbench para pruebas y consultas.
- Diagramación y diseño
    - UML realizado en MySQL Workbench (diagrama de clases y relación 1→1).
    - DER opcional para visualizar la estructura de la base.
- Documentación y referencias
    - Documentación oficial de Java JDBC.
    - Documentación oficial de MySQL.
    - Apuntes de la cátedra Programación 2 – UTN.
- Herramientas de apoyo
    - Microsoft Copilot (IA) para apoyo en redacción del informe, ejemplos de SQL y fragmentos de código.
    - Editor de texto (Google Docs) para la redacción del informe y exportación a PDF.

8. Links

Github: https://github.com/MaraBayurk/TUP-TPI-Programacion-II
Video: https://www.youtube.com/watch?v=HWVqiwCB4oQ