TECNICATURA UNIVERSITARIA EN
PROGRAMACIÓN A DISTANCIA

UTN
UNIVERSIDAD TECNOLÓGICA NACIONAL

MATERIA: Programación 2

COMISIÓN: 12

GRUPO: 109

FECHA: 17/11/2025

Apellido	Nombre	DNI	Rol
Pozo	Felipe	44.377.556	Diseño y Entidades
Barrios	Gonzalo	43.393.592	Base de Datos
Molteni	Emmanuel	36.873.560	DAO y Servicio
Gerbino	Facundo	43.018.365	AppMenu y Documentación

Trabajo Final Integrador – Programación 2

Sistema de Registro de Mascotas y Microchips

Introducción

Este trabajo final integrador tiene como objetivo desarrollar una base de datos **relacional** para el registro de mascotas y microchips. Partimos de un modelo inicial en el que se registran las mascotas con sus atributos principales: *// id - eliminado - nombre - especie - raza - fecha_nacimiento - duenio - microchip_id //* y la información del microchip: *// id - eliminado - código - fecha_implantacion - veterinaria - observaciones //.*

Además de la base de datos, el objetivo final fue desarrollar una aplicación en **Java** que modele las dos clases, relacionadas mediante una asociación unidireccional 1 a 1 en la que la clase "A" (mascota) referenciara la clase "B" (microchip), persistiendo los datos en la base relacional mediante **JDBC** y el patrón **DAO**, con operaciones transaccionales (commit/rollback) y un menú de consola **CLI** para gestionar el **CRUD**.

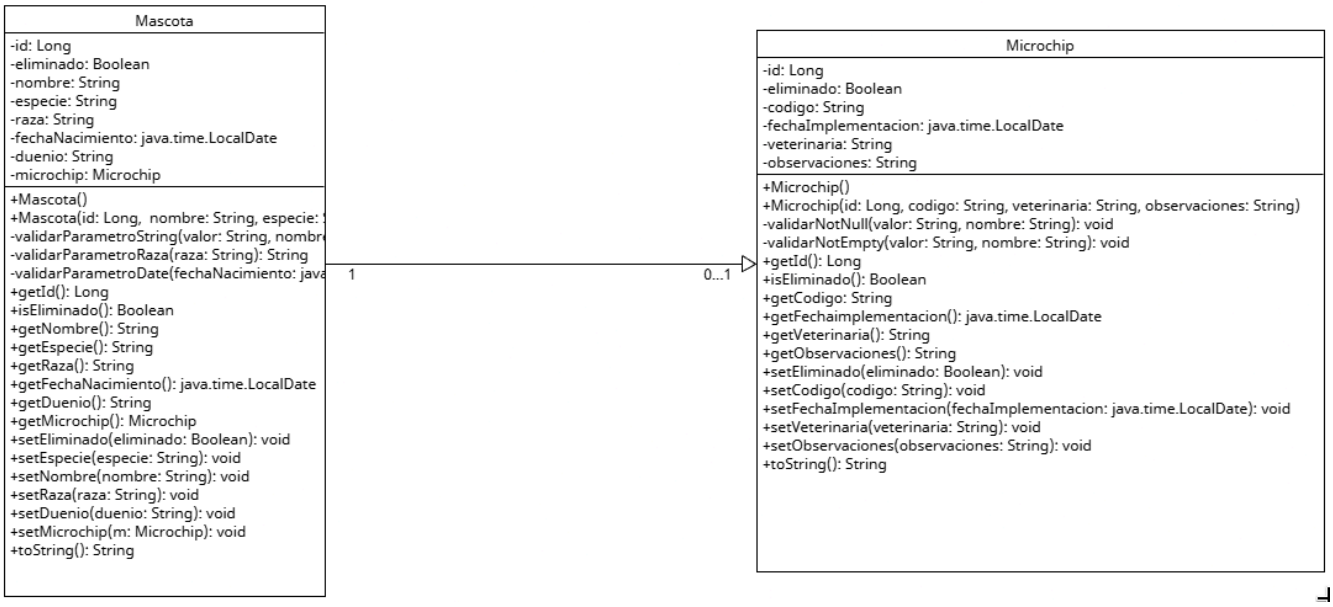
Elegimos el **dominio** Mascota > Microchip porque nos pareció interesante construir un sistema que podríamos ver implementado en la vida real, manejando la relación entre un animal y un almacenamiento de datos.

Índice

Etapas 1 y 2 – Diseño y Entidades

Para el trabajo integrador, decidimos simular el sistema de una veterinaria utilizando las tablas Mascota y Microchip, las cuales tienen una relación unidireccional desde A (Mascota) hacia B (Microchip), concluimos que era la mejor elección gracias a la experiencia adquirida en el **TPI** de Bases de Datos 1, el cual nos ayudó a entender mejor cómo se tendrían que manejar y enviar los datos a la base de datos.
Se decidió manejar el **CRUD** tanto para mascotas como para microchip aportando la capacidad de editar por separado cada entidad, como una forma de evitar errores de propagación. A su vez, decidimos que puedan asignar un microchip a una mascota que ya estuviera en el sistema, simulando un caso en el que el microchip se rompa.

Por último, creamos formas de filtrar mascotas y utilizamos el código de la clase microchip para localizar a la mascota a la que se quiere hacer algún cambio. Para manejar la relación, decidimos que mascota tenga como FK el id de Microchip, así evitamos errores en métodos donde sea necesario el cambio de Microchip.



Etapa 3 – Base de Datos (MySQL)

Creamos la base de datos en MySQLWorkbench haciendo una simple creación de tablas con las PK y FK para la relación unilateral entre Mascota > Microchip. Luego se ingresaron datos de prueba (INSERTS) para levantar el proyecto desde cero.

Para la conexión con Java se creó la clase **DatabaseConnection**, en la cual se ingresarán los datos necesarios (url, user, password) y se utilizara **DriverManager** para conectarlo a MySQL. Por último, se creó una clase Main **TestConnection** para que se intente una previa conexión de prueba y así arrancar con el proyecto con todo conectado sin problemas.

Creación de la base de datos y las tablas en MySQL respetando la relación Unidireccional:

```sql
CREATE DATABASE IF NOT EXISTS mascotas_db;

USE mascotas_db;

CREATE TABLE Microchip (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    eliminado BOOLEAN DEFAULT FALSE,
    codigo VARCHAR(25) NOT NULL UNIQUE,
    fecha_implantacion DATE,
    veterinaria VARCHAR(120),
    observaciones VARCHAR(255)
);

CREATE TABLE Mascota (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    eliminado BOOLEAN DEFAULT FALSE,
    nombre VARCHAR(60) NOT NULL,
    especie VARCHAR(30) NOT NULL,
    raza VARCHAR(60),
    fecha_nacimiento DATE,
    duenio VARCHAR(120) NOT NULL,
    microchip_id BIGINT UNIQUE,

    CONSTRAINT fk_mascota_microchip
        FOREIGN KEY (microchip_id)
        REFERENCES Microchip(id)
        ON DELETE CASCADE
);
```

```sql
USE mascotas_db;

INSERT INTO Microchip (codigo, fecha_implantacion, veterinaria, observaciones) VALUES
('ABC001', '2022-03-15', 'Veterinaria Central', 'Chip implantado sin incidencias.'),
('ABC002', '2021-11-20', 'Hospital Veterinario Sur', 'Revision anual, chip funcional.'),
('ABC003', '2023-01-01', 'Veterinaria Central', 'Nuevo microchip para cachorro.'),
('ABC004', '2020-06-01', 'Consultorio Dr. Mascotas', 'Mascota rescatada, se le implanto un chip.');

INSERT INTO Mascota (nombre, especie, raza, fecha_nacimiento, duenio, microchip_id) VALUES
('Fido', 'Perro', 'Labrador Retriever', '2019-05-10', 'Juan Perez', 1),
('Minina', 'Gato', 'Siames', '2020-08-22', 'Maria Lopez', 2),
('Rocky', 'Perro', 'Pastor Aleman', '2018-02-01', 'Carlos Gomez', 3),
('Burbuja', 'Gato', 'Balines', '2021-04-05', 'Laura Fernandez', 4);
```

```sql
SELECT * FROM Microchip;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	eliminado	codigo	fecha_implantacion	veterinaria	observaciones
1	0	ABC001	2022-03-15	Veterinaria Central	Chip implantado sin incidencias.
2	0	ABC002	2021-11-20	Hospital Veterinario Sur	Revision anual, chip funcional.
3	0	ABC003	2023-01-01	Veterinaria Central	Nuevo microchip para cachorro.
4	0	ABC004	2020-06-01	Consultorio Dr. Mascotas	Mascota rescatada, se le implanto un chip.
NULL	NULL	NULL	NULL	NULL	NULL

```sql
SELECT * FROM Mascota;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	eliminado	nombre	especie	raza	fecha_nacimiento	duenio	microchip_id
1	0	Fido	Perro	Labrador Retriever	2019-05-10	Juan Perez	1
2	0	Minina	Gato	Siames	2020-08-22	Maria Lopez	2
3	0	Rocky	Perro	Pastor Aleman	2018-02-01	Carlos Gomez	3
4	0	Burbuja	Gato	Balines	2021-04-05	Laura Fernandez	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Clase DatabaseConnection dentro de NetBeans para conectar Java con la base de datos:

```java
package config;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    // Ingresar cada uno su usuario, contraseña y direccion en donde tiene la base de datos
    private static final String url = "jdbc:mysql://127.0.0.1:3306/mascotas_db";
    private static final String user = "root";
    private static final String pass = "1234";

    // Metodo para el Driver
    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            throw new RuntimeException("Error: No se encontro el driver JDBC", e);
        }
    }

    // Metodo para conectarse con la Base de Datos
    public static Connection getConexion() throws SQLException {
        if (url == null || url.isEmpty() || user == null || user.isEmpty() || pass == null || pass.isEmpty()) {
            throw new SQLException("Configuracion de la base de datos invalida.");
        }
        return DriverManager.getConnection(url, user, pass);
    }
}
```

Clase Main TestConnection para hacer una conexión de prueba previa y resultado:

```java
package main;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import config.DatabaseConnection;
import java.util.Date;

public class TestConnection {
    public static void main(String[] args) {
        // Try que inicia la conexion con la Base de Datos llamando al metodo getConexion
        try (Connection conexion = DatabaseConnection.getConexion()) {

            // Condicional con instrucciones cuando conecte bien
            if (conexion != null) {
                System.out.println("Conexion con la BD establecida.");
```

```java
                // Llamado por SQL para traer la tabla mascota y mostrar informacion solo para verificar la conexion
                String sql = "SELECT * FROM mascota";
                try (PreparedStatement pstmt = conexion.prepareStatement(sql);
                     ResultSet rs = pstmt.executeQuery()) {
                    System.out.println("\nLista de mascotas: ");
                    while (rs.next()) {
                        int id = rs.getInt("id");
                        String nombre = rs.getString("nombre");
                        String especie = rs.getString("especie");
                        String raza = rs.getString("raza");
                        Date nacimiento = rs.getDate("fecha_nacimiento");
                        String duenio = rs.getString("duenio");
                        System.out.println(
                                "Id: " + id +
                                ". Nombre: " + nombre +
                                ". Especie: " + especie +
                                ". Raza: " + raza +
                                ". Fecha de Nacimiento: " + nacimiento +
                                ". Duenio: " + duenio + ".\n");
                    }
                }
            } else {
                System.out.println("No se pudo establecer la conexion.");
            }
        } catch (SQLException e) {
            System.err.println("Error al conectar a la BD: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

Prueba de funcionamiento:

```
Conexion con la BD establecida.

Lista de mascotas:
Id: 1. Nombre: Fido. Especie: Perro. Raza: Labrador Retriever. Fecha de Nacimiento: 2019-05-10. Duenio: Juan Perez.

Id: 2. Nombre: Minina. Especie: Gato. Raza: Siames. Fecha de Nacimiento: 2020-08-22. Duenio: Maria Lopez.

Id: 3. Nombre: Rocky. Especie: Perro. Raza: Pastor Aleman. Fecha de Nacimiento: 2018-02-01. Duenio: Carlos Gomez.

Id: 4. Nombre: Burbuja. Especie: Gato. Raza: Balines. Fecha de Nacimiento: 2021-04-05. Duenio: Laura Fernandez.
```

Etapas 4 y 5 – DAO y Service (Transacciones Obligatorias)

Se implementó una arquitectura multicapa clara que separa responsabilidades y facilita el mantenimiento del código.

Capa de Configuración (config)

- **Clase:** DatabaseConnection
- **Responsabilidad:** Gestión centralizada de conexiones a la base de datos MySQL
- **Funciones principales:**
    - Cargar el driver JDBC de MySQL
    - Proporcionar conexiones configuradas mediante el método getConexion()
    - Encapsular credenciales y URL de conexión

Capa de Entidades (entities)

- **Clases:** Mascota, Microchip
- **Responsabilidad:** Representar el modelo de dominio del negocio
- **Características:**
    - Contienen atributos que mapean directamente con las tablas de la base de datos
    - Implementan validaciones básicas de datos en setters
    - Manejan la relación unidireccional Mascota → Microchip
    - No tienen lógica de persistencia ni acceso a datos
    - Uso de tipos Java (LocalDate, Long, String) en lugar de tipos SQL

Capa de Acceso a Datos (dao)

- **Clases:** GenericDao (interfaz), MascotaDao, MicrochipDao
- **Responsabilidad:** Interacción directa con la base de datos
- **Funciones principales:**
    - Ejecutar operaciones CRUD (Create, Read, Update, Delete)
    - Mapear ResultSet a objetos Java
    - Construir y ejecutar sentencias SQL con PreparedStatement
    - **NO maneja transacciones** - recibe conexiones externas como parámetro
    - Métodos especializados de consulta (buscarPorCodigo, buscarPorMascotaId)

Capa de Servicios (service) Implementa el patrón DAO (Data Access Object) para abstraer la lógica de persistencia.

- **Clases:** GenericService, MascotaService, MicrochipService
- **Responsabilidad:** Lógica de negocio y gestión de transacciones
- **Funciones principales:**
    - Validar reglas de negocio antes de operaciones
    - Gestionar transacciones (commit/rollback)
    - Coordinar múltiples operaciones DAO en una sola transacción
    - Manejar conexiones (apertura, cierre, restauración de autoCommit)
    - Convertir excepciones técnicas en excepciones de negocio

El sistema implementa una jerarquía de validaciones en tres niveles aplicando el principio de validación en múltiples capas, garantizando la integridad de datos y la lógica de negocio.

La capa de excepciones implementa el patrón Service Layer para encapsular lógica de negocio usando las clases DaoException y ServiceException, con manejo estructurado de varios tipos de errores (SQL, conexion, logica de negocio y validación). Se implementó de una forma que permite propagar el contexto del error mediante causa encadenada.

```
1. Capa de Presentación (AppMenu)
   ├─ Validación de formato de entrada (números, fechas)
   ├─ Conversión de tipos
   └─ Mensajes de error amigables

2. Capa de Entidades (Mascota, Microchip)
   ├─ Validación de tipos de datos
   ├─ Validación de rangos y formatos
   └─ Lanzamiento de IllegalArgumentException

3. Capa de Servicio (MascotaService, MicrochipService)
   ├─ Validación de reglas de negocio
   ├─ Verificación de existencia en BD
   ├─ Validación de restricciones de integridad
   └─ Lanzamiento de ServiceException
```

Etapa 6 – AppMenu (Consola)

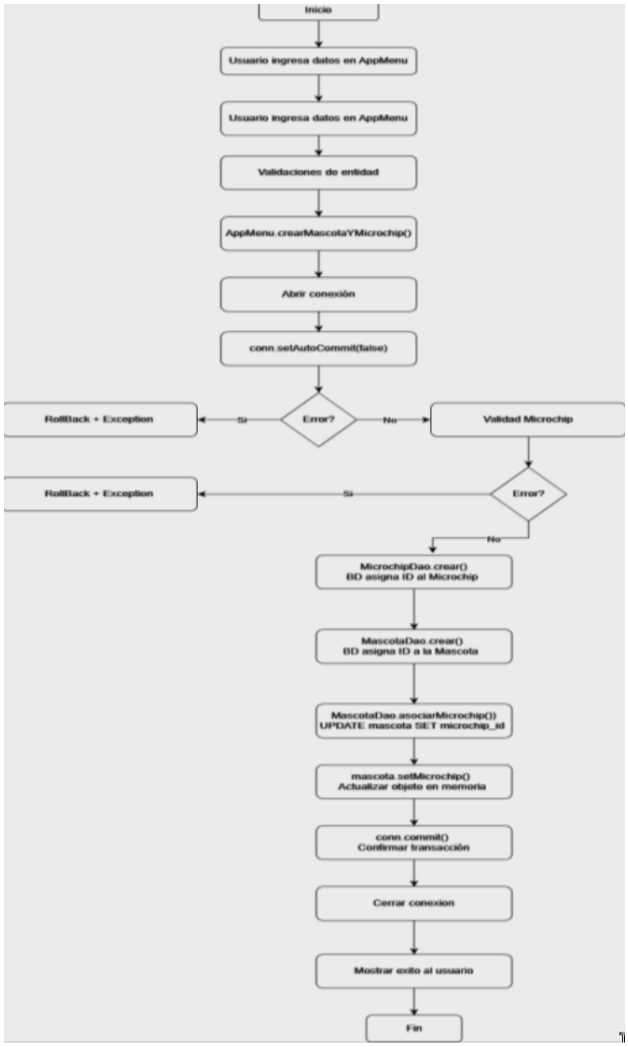
La clase AppMenu fue acoplada al main para actuar como la capa de presentación CLI. Se encarga de la interacción con el usuario mediante un menú navegable con una capa de seguridad que implementa conversión de entradas (String a Long, Dates, etc). El menú principal ofrece el CRUD completo para ambas entidades, la opción de asignación 1:1 directa, la operación transaccional compuesta y la búsqueda por código de microchip.
La implementación prioriza la robustez al capturar errores de formato y gestionar las excepciones de negocio mostrando mensajes específicos, que informan al usuario sobre fallos de validación, IDs inexistentes o problemas transaccionales (incluyendo el aviso de rollback aplicado).

Capa de Presentación (main)

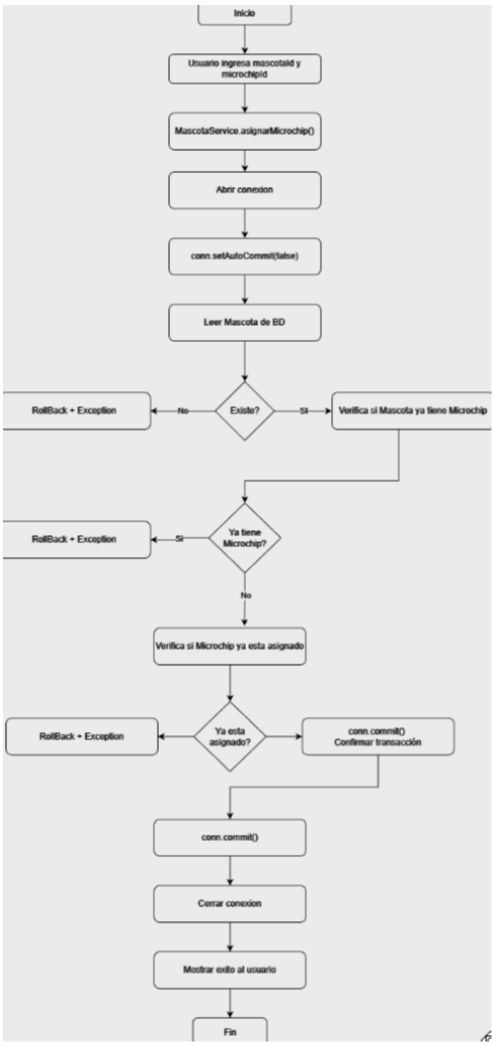
- **Clases:** Main, AppMenu, TestConnection
- **Responsabilidad:** Interfaz de usuario por consola (CLI)
- **Funciones principales:**
    - Presentar menús interactivos
    - Capturar entrada del usuario
    - Invocar servicios según las opciones seleccionadas
    - Mostrar resultados y mensajes de error
    - **NO contiene lógica de negocio ni acceso a datos**

Diagrama de Flujo

Creación de mascota con Microchip



Asignación de microchip existente



Herramientas y Fuentes

- **Lenguaje y Entorno:** Java Development Kit 21 (Oracle OpenJDK), Apache Netbeans IDE
- **Base de Datos:** MySQL Workbench y XAMPP
- **Librerías:** MySQL Connector (JDBC driver)
- **Diseño:** UMLETINO (para diagrama uml) y APPDiagrams para el diagrama de flujos