UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Final Integrador

Programación II

“Gestión de Mascotas y Microchips”

Comision 16

Baez, Santiago <*santiagobaezit@gmail.com*>

Marinoni, Macarena <*marinonimacarena@gmail.com*>

Valletto, Marianela <*mauvalletto@gmail.com*>

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional

Programación II

Profesor: Lobos, Diego

Tutor: Odiard, Andres

18/11/2025

ÍNDICE

Introducción

Este Trabajo Final Integrador consiste en el desarrollo de una aplicación en Java que implementa una **relación 1→1 unidireccional** entre Mascota y Microchip. El sistema se construyó siguiendo arquitectura por capas, con orientación a objetos, JDBC para la persistencia y buenas prácticas de diseño.

El proyecto se organiza en paquetes según su responsabilidad:

- config: manejo de la conexión a MySQL mediante JDBC.
- entities: definición del dominio (Mascota y Microchip) y su relación 1→1.
- dao: acceso a datos con el patrón DAO y operaciones CRUD seguras y parametrizadas.
- service: reglas de negocio, validaciones y manejo de transacciones (commit/rollback).
- main: menú de consola que integra todas las funcionalidades.

El sistema permite CRUD completo de mascotas y microchips, implementa baja lógica y garantiza la relación 1→1 mediante una clave foránea única. Las operaciones críticas, como crear mascota y microchip juntos, se gestionan mediante transacciones para asegurar atomicidad.

En síntesis, este proyecto integra los conceptos centrales de Programación II, aplicando Java, JDBC, arquitectura por capas y prácticas profesionales de desarrollo. El resultado es una aplicación robusta, ordenada y alineada con los estándares profesionales solicitados.

Integrantes y roles

El desarrollo fue realizado por tres integrantes con roles definidos:

- **Marianela Valletto**: diseño de la arquitectura, organización del proyecto, configuración del entorno y elaboración del informe final.
- **Macarena Marinoni**: análisis del dominio, modelado UML e implementación de entidades y DAOs con JDBC.
- **Santiago Báez**: capa de servicios, validaciones, reglas de negocio, operación transaccional principal y pruebas del sistema.

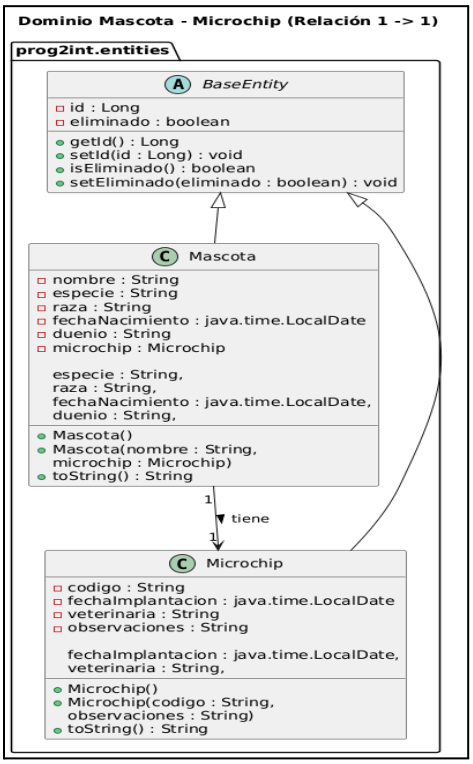
La distribución de tareas permitió un trabajo equilibrado y profesional, logrando una aplicación completa y bien estructurada.

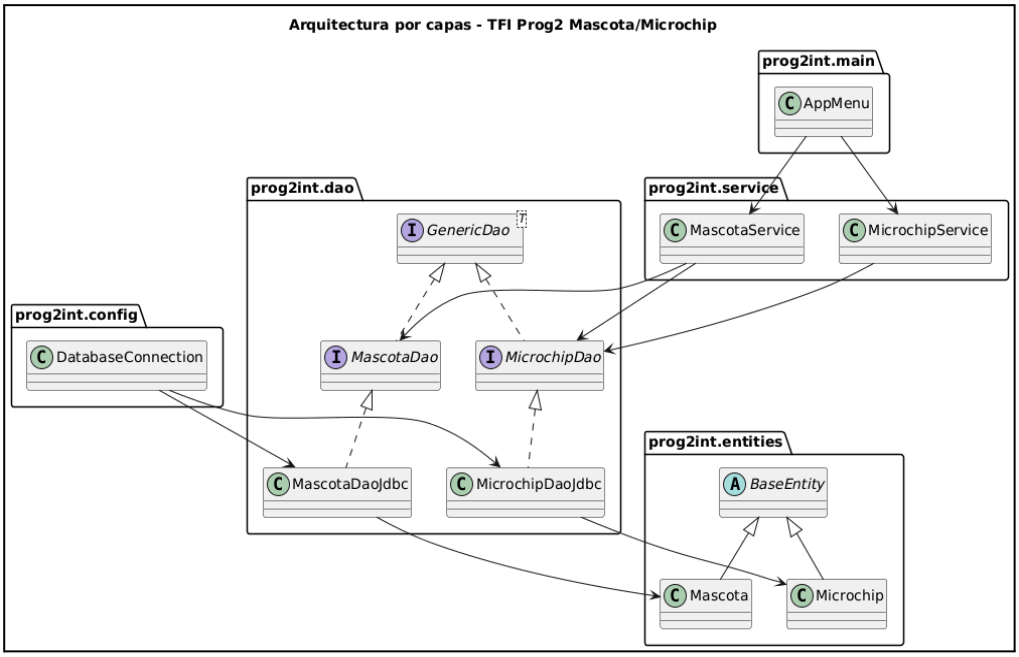
Diseño previo y modelado UML

El modelado conceptual inicial se realizó mediante un diagrama UML de clases que representa las estructuras fundamentales del sistema. Las entidades centrales del dominio son **Mascota** y **Microchip**, relacionadas mediante una asociación 1→1 unidireccional en la que la clase Mascota contiene una referencia opcional hacia Microchip. Esta decisión de diseño responde al funcionamiento natural del dominio: una mascota puede o no poseer un microchip implantado, pero un microchip siempre está asociado a una única mascota.

La clase **Mascota** incluye atributos característicos como el nombre, la especie, la raza, la fecha de nacimiento y el nombre del dueño, además del atributo compuesto que permite vincular, cuando corresponde, un objeto Microchip. La clase **Microchip** encapsula datos del dispositivo implantado, entre ellos el código único, la fecha de implantación, la veterinaria responsable y observaciones adicionales que puedan resultar relevantes. Ambas clases heredan los atributos id y eliminado a través de una clase base común, lo que garantiza consistencia en la identificación y en la gestión de bajas lógicas del sistema.

El diagrama UML también incluye la representación del patrón DAO mediante interfaces y sus implementaciones específicas, así como los servicios que consumen dichos DAOs. Este modelado previo fue fundamental para clarificar las responsabilidades, anticipar la estructura de paquetes y facilitar la implementación posterior con una base conceptual sólida y coherente.





Base de datos

La persistencia se implementó sobre MySQL utilizando un diseño relacional simple pero consistente. El modelo físico está compuesto por dos tablas: mascota y microchip. La relación 1⟶1 se materializa mediante una clave foránea única en la tabla microchip, representada por el atributo mascota_id. Esta restricción impide que dos microchips se asocien a la misma mascota y garantiza que cada mascota pueda poseer como máximo un único microchip.

La tabla mascota incluye atributos tales como nombre, especie, raza, fecha de nacimiento, dueño y el campo booleano eliminado, destinado a implementar la baja lógica. La tabla microchip contiene el código único, la fecha de implantación, la veterinaria, las observaciones y el campo mascota_id, declarado como UNIQUE para mantener la unicidad de la relación.

Para facilitar la reproducción del sistema, se elaboraron dos scripts SQL. El primero define la creación de la base de datos, la estructura de tablas, las claves primarias, la restricción única del código de microchip y la clave foránea con su respectiva regla de actualización. El segundo script carga datos de prueba adecuados para el testeo inicial del sistema. Este diseño asegura integridad referencial, coherencia entre el modelo conceptual y físico, y una interacción clara y eficiente con JDBC.

Arquitectura por capas

La arquitectura del proyecto se basa en el principio de separación de responsabilidades, distribuyendo los componentes en distintas capas lógicamente independientes.

La capa **config** centraliza toda la lógica relacionada con la obtención de conexiones hacia MySQL mediante JDBC. Su responsabilidad es leer el archivo de configuración con las credenciales necesarias y proporcionar un método estático que devuelva conexiones listas para usar.

La capa **entities** modela el dominio y agrupa las clases Mascota y Microchip. Ambas clases contienen atributos privados, métodos accesores y constructores adecuados que facilitan la instancia y manipulación de objetos. La inclusión del atributo eliminado permite sustituir la eliminación física por la baja lógica, reduciendo riesgos de pérdida de datos.

La capa **dao** implementa el acceso a datos mediante el patrón DAO, lo que favorece el desacoplamiento y la flexibilidad del sistema. Las interfaces definen operaciones genéricas como crear, leer, actualizar y eliminar, mientras que las implementaciones concretas contienen consultas

SQL parametrizadas que interactúan con la base de datos. Cada DAO se responsabiliza únicamente de su entidad, evitando fugas de lógica hacia otras capas.

La capa **service** es un componente intermedio que incorpora reglas de negocio y validaciones necesarias para evitar inconsistencias en los datos. Además, coordina la interacción entre DAOs, especialmente en las operaciones que requieren transacciones para garantizar la integridad.

Finalmente, la capa **main**, a través de la clase AppMenu, constituye la interfaz textual para el usuario. Este menú organiza las distintas funcionalidades y delega toda la lógica en la capa de servicios, asegurando una estructura limpia y sencilla para la interacción.

Persistencia, transacciones y commit/rollback

La persistencia del sistema se implementa mediante JDBC utilizando consultas SQL parametrizadas que garantizan seguridad y eficiencia. Para mantener la integridad de los datos, especialmente durante operaciones compuestas, se emplean transacciones mediante el uso explícito de setAutoCommit(false).

La operación crítica del sistema es el registro conjunto de una mascota y su microchip. Esta operación involucra dos inserciones dependientes: una en la tabla mascota y otra en la tabla microchip. Para garantizar la atomicidad de esta acción, se utiliza una única conexión compartida y se aplican los métodos commit() o rollback(), según el éxito o fracaso de las operaciones.

Si la inserción del microchip falla (por ejemplo, por un código duplicado), la transacción completa se revierte y no se registra la mascota, evitando inconsistencias y microchips huérfanos. Este uso explícito de transacciones demuestra una comprensión profunda del manejo de integridad a nivel de capa de servicio y es uno de los elementos centrales requeridos en el trabajo práctico.

Validaciones y reglas de negocio

La capa de servicios incorpora distintas validaciones que aseguran la calidad y consistencia de los datos. Entre ellas se incluyen la verificación de que los campos obligatorios no se encuentren vacíos, la conversión a mayúsculas para estandarizar los datos de entrada, la comprobación de unicidad del código de microchip y la imposibilidad de registrar un microchip sin una mascota asociada.

La elección de realizar estas validaciones en la capa de servicios y no en la capa DAO responde al criterio de separar responsabilidades y evitar mezclar lógica de negocio con lógica de persistencia. De

esta forma, se mantiene un diseño limpio y coherente con las buenas prácticas de la arquitectura por capas.

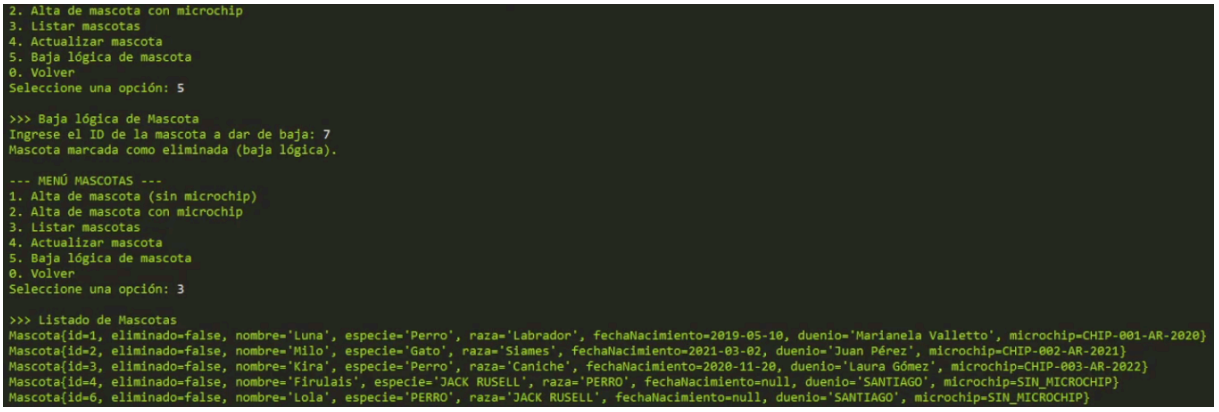
Pruebas realizadas

Las pruebas del sistema incluyeron tanto la ejecución de operaciones desde el menú principal como la verificación directa en la base de datos mediante consultas SQL. Se evaluaron altas, modificaciones, lecturas y bajas lógicas tanto para mascotas como para microchips. Asimismo, se realizaron pruebas específicas sobre la operación transaccional, verificando el correcto funcionamiento del rollback frente a microchips duplicados o entradas inválidas.

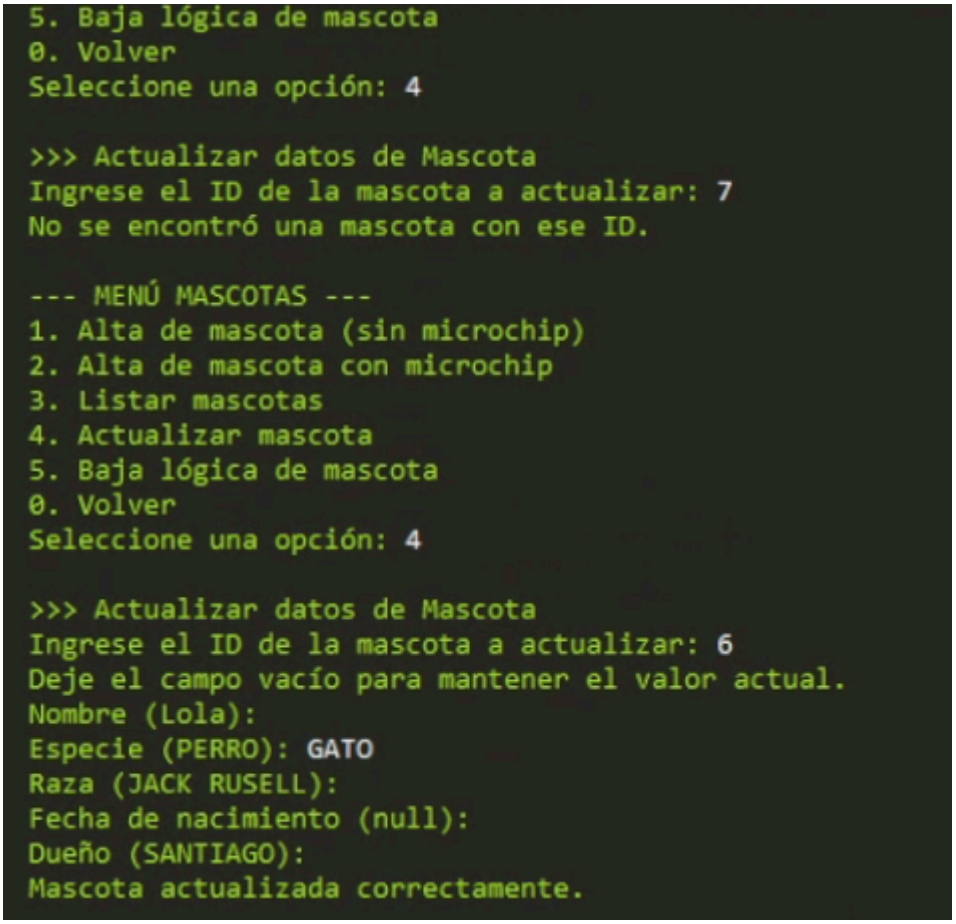
También se corroboró el correcto funcionamiento del listado completo, las búsquedas avanzadas por nombre o dueño, la lectura por ID, la actualización de registros y la correcta conversión de texto a mayúsculas en las operaciones ingresadas por consola. Finalmente, se validó mediante consultas SQL la consistencia de la relación 1⟶1 en la base de datos.

ALTA

BAJA



MODIFICACIÓN



Conclusión

La implementación de este Trabajo Final Integrador permitió consolidar los conocimientos adquiridos en Programación II, integrando conceptos de programación orientada a objetos, diseño por capas, patrón DAO, validaciones, manejo de excepciones y persistencia mediante JDBC. Además, proporcionó una experiencia práctica en la gestión de transacciones y en la aplicación de estrategias para garantizar la integridad de los datos.

La relación 1→1 entre Mascota y Microchip se implementó correctamente mediante el uso de clave foránea única, un diseño que se mantuvo consistente tanto en el modelo conceptual como en la base de datos y en la estructura del código. La organización modular del proyecto facilitó su mantenimiento y escalabilidad, y la utilización de GitHub aportó un entorno profesional de trabajo colaborativo.

El sistema desarrollado es robusto, claro y escalable, permitiendo futuras extensiones como la incorporación de nuevas entidades, la implementación de una interfaz gráfica o la exposición de servicios web. En conjunto, este proyecto constituye una integración sólida de técnicas y conocimientos fundamentales para avanzar hacia desarrollos más complejos y profesionales.

Mejoras futuras

El sistema desarrollado sienta una base sólida para futuras ampliaciones. Entre las mejoras posibles se encuentran la incorporación de nuevas entidades como consultas veterinarias, historial de vacunación o datos del dueño, lo que permitiría convertir la aplicación en un sistema de gestión más completo.

Otra evolución natural sería reemplazar el menú de consola por una interfaz gráfica con JavaFX o Swing, mejorando la experiencia de usuario e incorporando funciones adicionales, como la generación de reportes en PDF o Excel.

A nivel avanzado, el proyecto podría evolucionar hacia una aplicación distribuida mediante servicios REST con Spring Boot, habilitando clientes web o móviles. También sería conveniente integrar logging, pruebas automatizadas con JUnit y autenticación, elevando el nivel profesional del sistema.

Fuentes y herramientas utilizadas

Para el desarrollo del proyecto se emplearon herramientas tecnológicas y recursos de consulta, detallados a continuación junto con sus versiones y enlaces oficiales:

Herramientas de programación

- Java SE 24 (LTS): https://www.oracle.com/java/technologies/javase-downloads.html
- Apache NetBeans IDE 24: https://netbeans.apache.org/
- JDBC – Driver MySQL Connector/J 8.3: https://dev.mysql.com/downloads/connector/j/
- MySQL Server 8.0: https://dev.mysql.com/downloads/mysql/
- MySQL Workbench 8.0: https://dev.mysql.com/downloads/workbench/

Control de versiones

- GitHub – Gestión del repositorio remoto:
  https://github.com/MaruValletto/TFI_Prog2_MascotaMicrochip.git

Modelado UML

- PlantUML (versión online): https://plantuml.com/

Documentación y recursos teóricos

- Oracle Documentation (JDBC, POO, patrones y buenas prácticas): https://docs.oracle.com/en/java/
- Bibliografía recomendada por la cátedra: Videos Tutoriales, Filminas con teoría, entre otros.

Inteligencia Artificial utilizada

- ChatGPT – OpenAI: https://chat.openai.com/
  Utilizado como apoyo complementario para mejora de redacción, organización del informe, ejemplos conceptuales y revisión de estilo y coherencia.

Comentario sobre la conformación del equipo

El trabajo es presentado solo por tres integrantes, a pesar de que la consigna establecía un equipo de cuatro personas. Inicialmente, el grupo estaba constituido por cuatro estudiantes; sin embargo, durante los últimos días previos a la entrega, se perdió toda comunicación con uno de los integrantes no participo de las actividades de coordinación ni de la realización final del trabajo. Esta situación se replicó también en otras asignaturas, dificultando la posibilidad de integrar sus aportes o reorganizar las responsabilidades a tiempo.

En consecuencia, las tareas fueron redistribuidas entre los tres integrantes restantes, quienes completaron el trabajo en su totalidad para cumplir con los plazos establecidos.