TECNICATURA UNIVERSITARIA EN
PROGRAMACIÓN A DISTANCIA

PROGRAMACIÓN II
Trabajo Final Integrador

Tema: DispositivoIoT → ConfiguracionRed

Alumno :

- López Francisco
- Luna Gonzalo
- Malatesta Juan Ignacio
- González Federico

Profesor :

- Enferrel Ariel

Tutor :

- Ferro Tomás

GitHub:
https://github.com/fedeglz/TFI_Prog2_DispositivoIoT_ConfigRed

Video: https://www.youtube.com/watch?v=FHW30OzRqNg

1. Integrantes y Roles del Proyecto

Este trabajo fue desarrollado por un equipo de cuatro integrantes, donde cada uno asumió un rol específico para cubrir las diferentes capas y funcionalidades del sistema:

-**Francisco López:** Presentador principal y encargado de la visión general del sistema.
-**Gonzalo Luna:** Responsable de entidades del dominio y acceso a base de datos.
-**Juan Ignacio Malatesta**: Encargado de la capa de servicios y manejo de reglas de negocio.
-**Federico González:** Encargado de la interacción del usuario y demostración en vivo.

2. Elección del Dominio y Justificación

El dominio seleccionado fue la gestión de dispositivos IoT y sus configuraciones de red, el cual nos resultó adecuado para aplicar los contenidos principales de la materia.

Elegimos este dominio por los siguientes motivos:
-Es un campo actual y relevante en el área de sistemas y telecomunicaciones.
-Permite trabajar con objetos que representan entidades del mundo real (sensores, actuadores, routers, etc.).
-La relación 1 a 1 entre dispositivo y configuración de red es ideal para practicar vínculos entre entidades y aplicar claves foráneas en bases de datos relacionales.
-Facilita la implementación de un CRUD completo que involucra varias capas.
-Permite aplicar conceptos de Programación Orientada a Objetos, arquitectura por capas y persistencia con JDBC, objetivos fundamentales del trabajo.

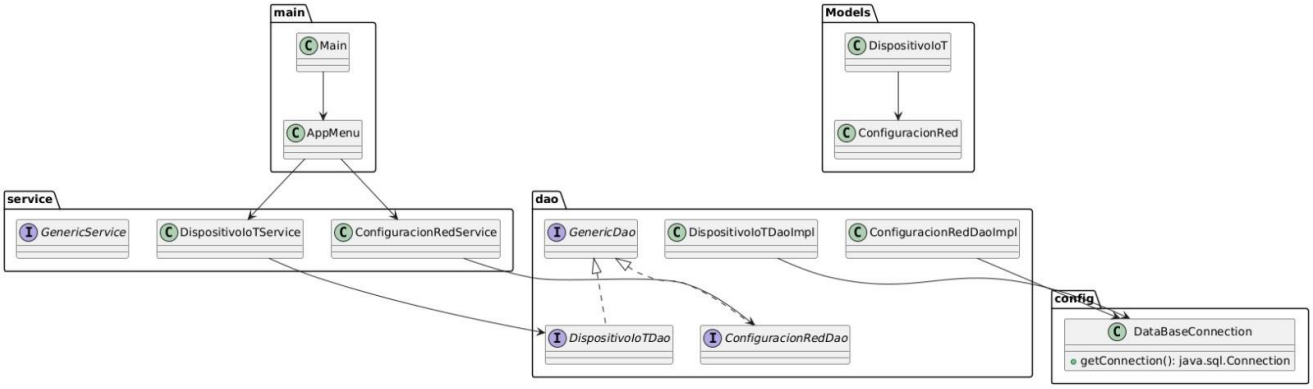
3. Diseño del Sistema

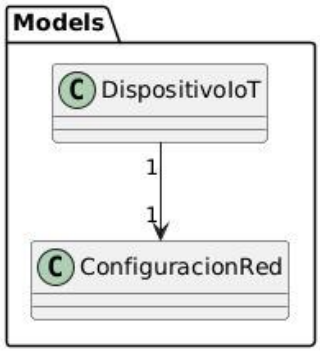
3.1 Relación 1 a 1 y decisiones de diseño

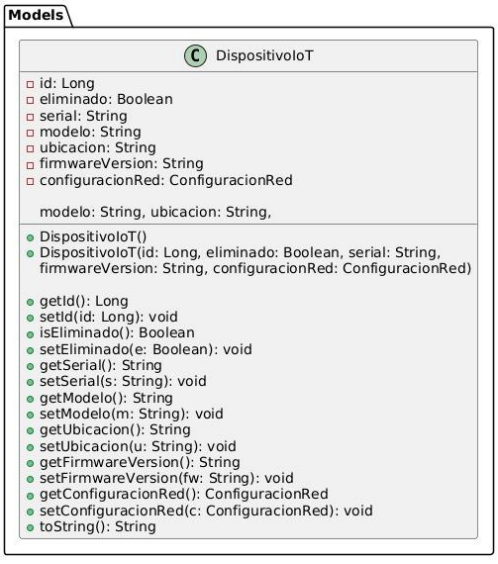
Una de las decisiones más importantes del diseño fue la relación uno a uno entre DispositivoIoT y ConfiguracionRed. Para implementarla, evaluamos dos alternativas:

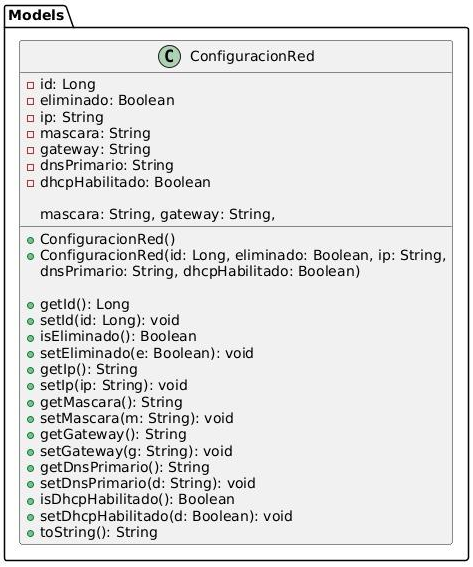
-Clave primaria compartida (PK = FK)
-Clave foránea única (FK UNIQUE)
Finalmente se optó por utilizar una clave foránea única en la tabla de dispositivos (configuracion_id), ya que simplifica el CRUD, evita dependencias fuertes en el DAO y facilita la creación independiente de ambos registros.

3.2 UML del proyecto









4. Arquitectura por Capas

El proyecto fue desarrollado siguiendo una estructura de 5 capas, distribuidas en diferentes paquetes:

1) Paquete Config

-Contiene la clase encargada de establecer la conexión con MySQL mediante JDBC.
-Centraliza parámetros de conexión y evita duplicación de código.

2) Paquete Models

-Contiene las entidades del sistema (DispositivoIoT y ConfiguracionRed).
-Define atributos, constructores y métodos de acceso.
-Representa la capa de dominio.

3) Paquete Dao

-Contiene las clases DAO responsables del acceso a la base de datos.
-Implementa CRUD utilizando JDBC y PreparedStatement.
-Maneja consultas SQL, excepciones y transformaciones entre objetos y registros.

4) Paquete Service

-Contiene la lógica de negocio del sistema.
-Realiza validaciones y coordinaciones entre DAOs.
-Implementa una función transaccional con manejo de commit y rollback.

5) Paquete Main (interfaz)

-Contiene el menú principal del sistema.
-Permite al usuario ejecutar las operaciones CRUD desde consola.

5. Persistencia: Estructura de la Base de Datos y Transacciones

5.1 Estructura de tablas

Utilizamos dos tablas principales:

Tabla configuracion_red

-id (PK)
-ip
-mascara
-gateway
-dns
-activo (baja lógica)

Tabla dispositivo_iot

-id (PK)
-nombre
-tipo
-configuracion_id (FK UNIQUE → configuracion_red.id)
-activo

5.2 Orden de operaciones (lógica de persistencia)

1- Insertar primero la configuración de red.
2- Obtener el ID generado automáticamente.
3- Crear el dispositivo utilizando ese ID.
4- Incluir baja lógica para evitar pérdida de datos históricos.

5.3 Transacciones en el sistema

El método pruebaRollbackTransaccional() demuestra el uso de transacciones manuales:

-Se desactiva el auto-commit.
-Se realizan dos inserciones consecutivas.
-Si la segunda inserción falla, se activa un rollback para mantener la integridad de los datos.

Esta prueba verifica que el sistema es capaz de evitar estados inconsistentes.

La prueba de rollback se ejecuta desde el método pruebaRollbackTransaccional() en la clase Service, mostrando en consola la reversión ante un error intencional.

6. Validaciones y Reglas de Negocio

El sistema implementa varias validaciones para mantener coherencia y datos correctos:

-Validación de campos obligatorios como nombre de dispositivo o dirección IP.
-Verificación de que una configuración exista antes de asociarla a un dispositivo.
-Restricción de eliminación utilizando baja lógica (activo = 0).
-Prevención de operaciones inválidas como editar una entidad inactiva.

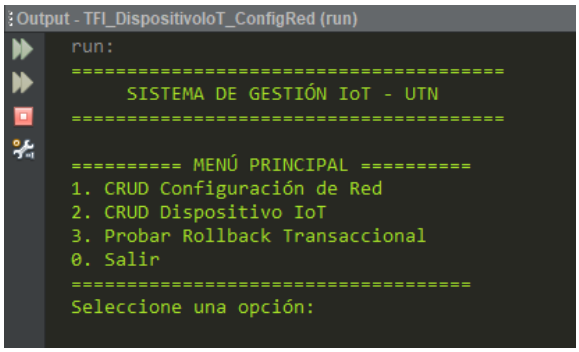
Estas validaciones se encuentran principalmente en la capa Service.

7. Pruebas Realizadas

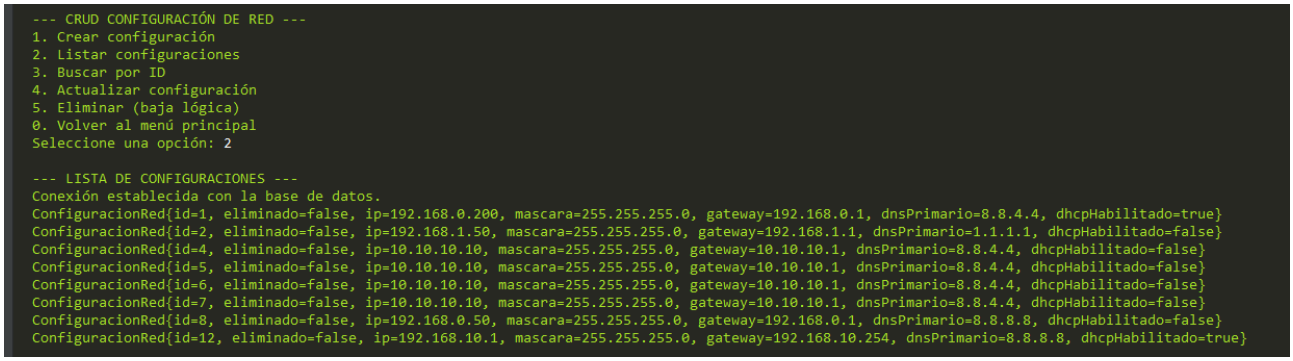
Durante la validación del sistema realizamos las siguientes pruebas:

-Crear una configuración de red.
-Crear un dispositivo IoT asociado.
-Listar configuraciones y dispositivos activos.
-Actualizar datos de red.
-Eliminar mediante baja lógica.
-Probar la relación 1 → 1 mediante una consulta SQL con JOIN.
-Ejecutar la prueba de rollback mostrando en consola el funcionamiento transaccional.

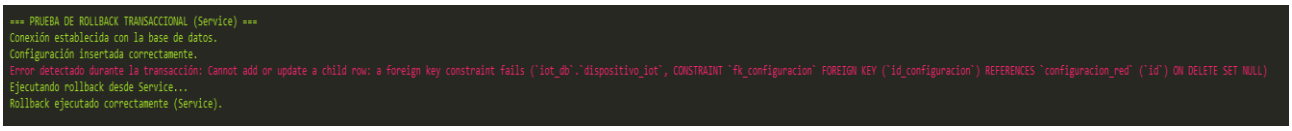
Ejecución del menú principal del sistema



Inserción y listado de configuraciones de red



Inserción de dispositivo IoT asociado a una configuración



Rollback transaccional ante error de integridad

```
--- CRUD DISPOSITIVO IoT ---
1. Crear dispositivo
2. Listar dispositivos
3. Buscar por ID
4. Actualizar dispositivo
5. Eliminar (baja lógica)
0. Volver al menú principal
Seleccione una opción: 2

--- LISTA DE DISPOSITIVOS ---
Conexión establecida con la base de datos.
DispositivoIoT{id=1, eliminado=false, serial=SN-0001, modelo=ESP32-WROOM, ubicacion=Oficina Principal, firmwareVersion=v1.2.0, configuracionRed=192.168.0.200}
DispositivoIoT{id=2, eliminado=false, serial=SN-0002, modelo=Raspberry Pi 4, ubicacion=Laboratorio A, firmwareVersion=v2.0.1, configuracionRed=192.168.1.50}
DispositivoIoT{id=3, eliminado=false, serial=SN-0003, modelo=Arduino Nano 33, ubicacion=Depósito, firmwareVersion=v1.0.5, configuracionRed=10.10.10.5}
```

Consulta SQL con JOIN entre las tablas

```
Limit to 1000 rows
1  SELECT d.id, d.modelo, c.ip, c.dhcp_habilitado
2  FROM dispositivo_iot d
3  JOIN configuracion_red c ON d.id_configuracion = c.id;
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	modelo	ip	dhcp_habilitado
1	ESP32-WROOM	192.168.0.200	1
2	Raspberry Pi 4	192.168.1.50	0
3	Arduino Nano 33	10.10.10.5	1
7	ESP32-CAP-PRO	10.10.10.10	0

8. Conclusiones

El desarrollo de este TP Integrador nos permitió aplicar los conceptos clave aprendidos durante la materia Programación II. Entre los aprendizajes principales se encuentran:

-Uso correcto de Programación Orientada a Objetos.
-Implementación de una arquitectura por capas clara y mantenible.
-Manejo de una base de datos MySQL utilizando JDBC.
-Trabajo con relaciones 1 a 1 e integridad referencial.
-Implementación de transacciones para garantizar consistencia.
-Desarrollo de un CRUD funcional por consola.

En general, el proyecto nos permitió comprender la importancia de estructurar un sistema de forma profesional y de separar responsabilidades.

9. Mejoras Futuras

Algunas mejoras posibles para versiones futuras del sistema son:

-Incorporar una interfaz gráfica con JavaFX o Swing.
-Implementar validaciones más avanzadas para direcciones IP.
-Agregar logs de auditoría.
-Exportar configuraciones a JSON o XML.
-Integración con APIs o dispositivos IoT reales.

10. Fuentes y Herramientas Utilizadas

Herramientas:
-NetBeans
-MySQL Workbench
-Driver JDBC
-GitHub
-draw.io para UML
-Consola de Windows/Linux

Fuentes consultadas:
-Documentación oficial de JDBC (Oracle)
-Manual de MySQL
-Apuntes de la cátedra

Uso de Inteligencia Artificial:

Se utilizó IA (ChatGPT) como apoyo para organización del contenido, manteniendo siempre criterio propio y validación manual del código y diseño.