Trabajo Practico Integrador — Programación 2

Alumnos: Alliot Fermin, Antuña Gabriel, Camilo Quiroga y Diego Pavon
Carrera: Tecnicatura Universitaria en Programación
Materia: Programación 2
Dominio: Producto → CodigoBarras (1→1 unidireccional)
Fecha: 17/11/2025

1) Integrantes y roles

- **Fermín Alliot**, **Gabriel Antuña, Camilo Quiroga y Diego Pavon** trabajamos **en conjunto en todas las actividades**, utilizando una modalidad de **pair programming** y revisión cruzada.
- **Responsabilidades compartidas:**
    - Diseño de dominio y decisiones de arquitectura.
    - Modelado de base de datos y preparación de scripts SQL (creación, seed y verificación).
    - Implementación en Java con JDBC (patrón DAO + Service).
    - Desarrollo del **AppMenu** (CLI) con CRUD completo y búsqueda por campo clave.
    - Pruebas (casos felices, bordes y **rollback**).
    - Documentación (README, informe) y grabación del video de defensa.

---

2) Elección del dominio y justificación (Código de barras ↔ Producto)

Justificación: Es un escenario realista y frecuente en retail/stock, donde cada **Producto** puede tener **un único Código de Barras** (EAN/UPC). La relación **1→1 unidireccional** es adecuada: **Producto** conoce (y referencia) a **CodigoBarras**, pero **CodigoBarras** no referencia de vuelta.
Beneficios:

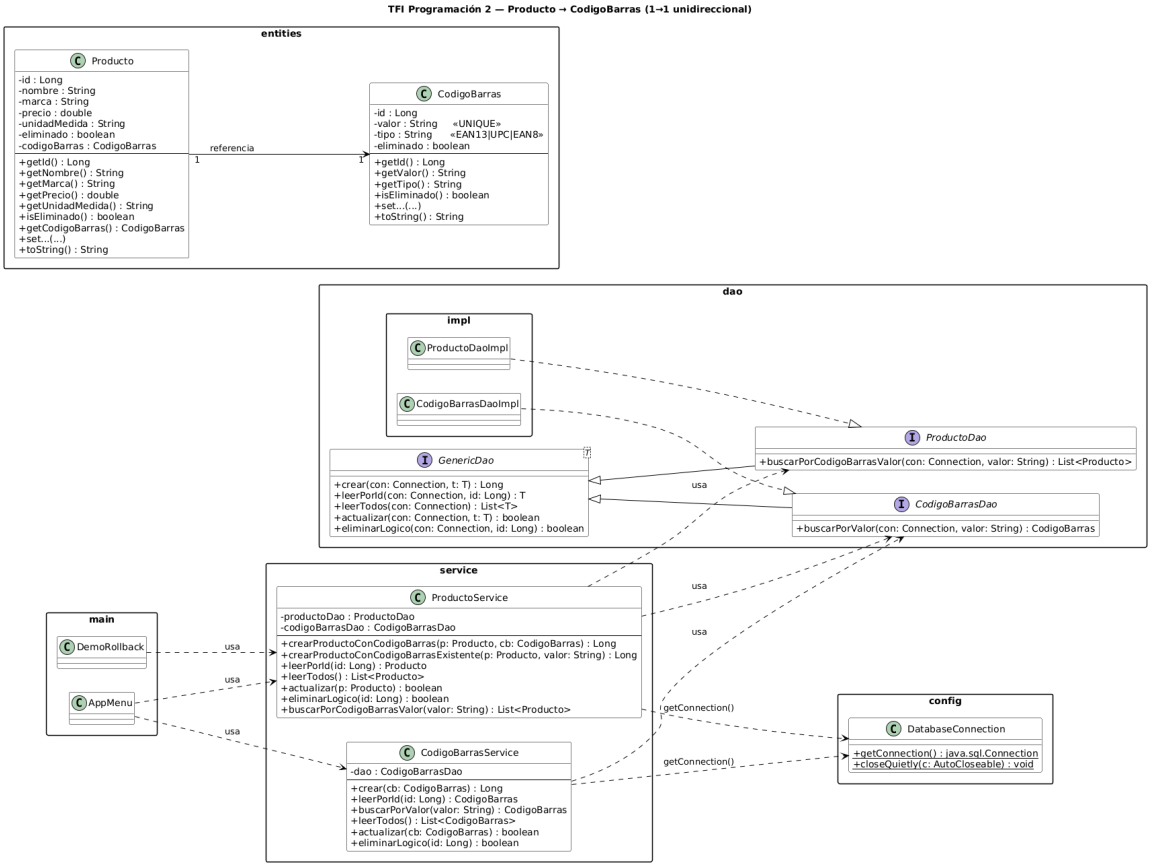
- Búsqueda natural por **valor** (EAN/UPC).
- Integridad conceptual simple: un código no se comparte entre productos.

- Práctico para demostrar **transacciones** y **restricciones de unicidad** en la BD.

---

3) Diseño: decisiones clave (1→1, FK única vs PK compartida) + UML

- **Tipo de relación: 1→1 unidireccional** (**A=Producto → B=CodigoBarras**).
- **Decisión principal (1→1):** se implementó con **FK única** en Producto.codigo_barras_id.
    - **Ventajas:** desacoplamiento, claridad, flexibilidad para tener productos sin código asociado (si el negocio lo permite), y menor complejidad de migración.
- **Alternativa considerada: PK compartida** (misma PK en ambas tablas).
    - **Motivo de descarte:** acopla fuertemente los ciclos de vida y complica escenarios donde el producto puede existir previo a asignar un código.
- **UML** (resumen):
    - Producto contiene un atributo privado CodigoBarras codigoBarras.
    - **Cardinalidad:** Producto 1 --> 1 CodigoBarras (solo **Producto** conoce a **CodigoBarras**).



Clases (atributos relevantes):

- CodigoBarras { id, valor(UNIQUE, len=13, dígitos), tipo(EAN13|UPC|EAN8), eliminado }
- Producto { id, nombre, marca, precio(>=0), unidadMedida, eliminado, codigoBarras }

---

4) Arquitectura por capas (responsabilidades de cada paquete)

- **config/**
  - DatabaseConnection: lectura de propiedades y entrega de Connection JDBC (MySQL).
- **entities/**
  - POJOs: Producto, CodigoBarras. Constructores, getters/setters y toString() **no recursivo**.
- **dao/**

  - Interfaces (GenericDao, ProductoDao, CodigoBarrasDao) y **implementaciones JDBC con PreparedStatement**.
  - Todos los métodos **aceptan Connection externa** para poder compartir la misma conexión en una transacción.

- **service/**
  - Reglas de negocio, **validaciones** y **transacciones**.
  - **Orden atómico** en alta compuesta: *crear B → asociar en A → crear A*.
  - commit() si todo OK. rollback() ante cualquier excepción.
- **main/**
  - AppMenu (CLI): CRUD de A y B, **búsqueda por CodigoBarras.valor**, **eliminación lógica** y **opción de demo de rollback**.

---

5) Persistencia: estructura de la base, orden de operaciones y transacciones

Estructura (MySQL 8):

- **Tabla CodigoBarras**
  - id (PK, BIGINT AI)
  - valor (CHAR(13), **UNIQUE**, CHECK longitud=13 y dígitos)
  - tipo (ENUM('EAN13','UPC','EAN8'), default 'EAN13')
  - eliminado (TINYINT(1), default 0)
- **Tabla Producto**
  - id (PK, BIGINT AI)
  - nombre (VARCHAR(120) NOT NULL)
  - marca (VARCHAR(80))
  - precio (DECIMAL(12,2) NOT NULL, >= 0 validado en Service)
  - unidad_medida (VARCHAR(16), default 'unidad')
  - eliminado (TINYINT(1), default 0)

- codigo_barras_id (BIGINT, **UNIQUE**, FK → CodigoBarras.id)

Orden atómico en Service (alta de Producto + Código NUEVO):

1. setAutoCommit(false) sobre una **misma Connection compartida**.
2. **Crear CodigoBarras (B)** → obtener id.
3. **Asociar** el CodigoBarras recién creado al **Producto** (codigo_barras_id).
4. **Crear Producto (A)**.
5. commit() si todo OK.
6. En caso de excepción (por ejemplo, UNIQUE): **rollback()** y **no queda nada a medias**.
7. Restaurar autoCommit(true) en finally y cerrar recursos.

Commit/Rollback:

- **Dónde:** dentro de ProductoService (método transaccional).
- **Cuándo:** commit() al finalizar los 3 pasos sin errores, rollback() ante cualquier excepción.

```
--- exec:3.1.0:exec (default-cli) @ tfi-programacion2 ---

=== TFI Programación 2 — Producto → CodigoBarras (1→1) ===
1) Alta CodigoBarras
2) Alta Producto + CodigoBarras NUEVO (transacción)
3) Alta Producto usando CodigoBarras EXISTENTE (transacción)
4) Listar Productos
5) Buscar Producto por valor de CodigoBarras
6) Eliminar lógico Producto
7) Listar Codigos de Barras
8) Eliminar lógico CodigoBarras
9) Demostración de rollback (UNIQUE valor)
0) Salir
Opción: 9
Forzando violación UNIQUE con valor CB=7791234567890
Rollback OK — Excepción esperada: Duplicate entry '7791234567890' for key 'codigobarras.uq_codigobarras_valor'

=== TFI Programación 2 — Producto → CodigoBarras (1→1) ===
1) Alta CodigoBarras
2) Alta Producto + CodigoBarras NUEVO (transacción)
3) Alta Producto usando CodigoBarras EXISTENTE (transacción)
4) Listar Productos
5) Buscar Producto por valor de CodigoBarras
6) Eliminar lógico Producto
7) Listar Codigos de Barras
8) Eliminar lógico CodigoBarras
9) Demostración de rollback (UNIQUE valor)
0) Salir
Opción: 0
Fin.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  39.915 s
Finished at: 2025-11-16T15:24:00-03:00
------------------------------------------------------------------------
```

---

6) Validaciones y reglas de negocio

- **Formato de CodigoBarras.valor:** 13 dígitos (validado por **DB** con CHECK y por Service a nivel de entradas).
- **Unicidad:**
    - CodigoBarras.valor **UNIQUE**.
    - Producto.codigo_barras_id **UNIQUE** (garantiza 1→1 efectivo).
- **Precio de Producto:** no puede ser negativo (validado en Service).
- **Eliminación lógica:** campo eliminado en ambas entidades para **no borrar datos** y permitir auditoría básica.
- **Búsqueda clave (AppMenu):** por CodigoBarras.valor (EAN/UPC).

---

7) Pruebas realizadas (capturas del menú y consultas SQL útiles)

Casos felices:

- Alta de **CodigoBarras** válido (longitud 13 y dígitos).
- Alta de **Producto + CB nuevo** (transacción OK).
- Alta de **Producto con CB existente** (transacción OK).
- **Listados** y **búsqueda por valor**.
- **Eliminación lógica** de Producto y de CodigoBarras.

Casos de borde / error:

- valor vacío, no numérico o longitud ≠ 13 → rechazo con mensaje claro.
- precio < 0 → rechazo en Service.
- Intento de duplicar valor → **violación de UNIQUE** y **rollback**.
- Intento de reutilizar el mismo codigo_barras_id en 2 productos → **violación de UNIQUE**.

Consultas de verificación (03_verify.sql):

- Duplicados de valor en CodigoBarras:
- SELECT valor, COUNT(*) c FROM CodigoBarras GROUP BY valor HAVING c > 1;
- Verificación 1→1 (mismo codigo_barras_id en >1 producto):
- SELECT codigo_barras_id, COUNT(*) c
- FROM Producto
- WHERE codigo_barras_id IS NOT NULL
- GROUP BY codigo_barras_id
- HAVING c > 1;
- Conteos y join de integridad:
- SELECT (SELECT COUNT(*) FROM CodigoBarras) cb_count,
- (SELECT COUNT(*) FROM Producto) p_count;
- SELECT p.id, p.nombre, cb.valor, cb.tipo
- FROM Producto p LEFT JOIN CodigoBarras cb ON p.codigo_barras_id = cb.id

- ORDER BY p.id;

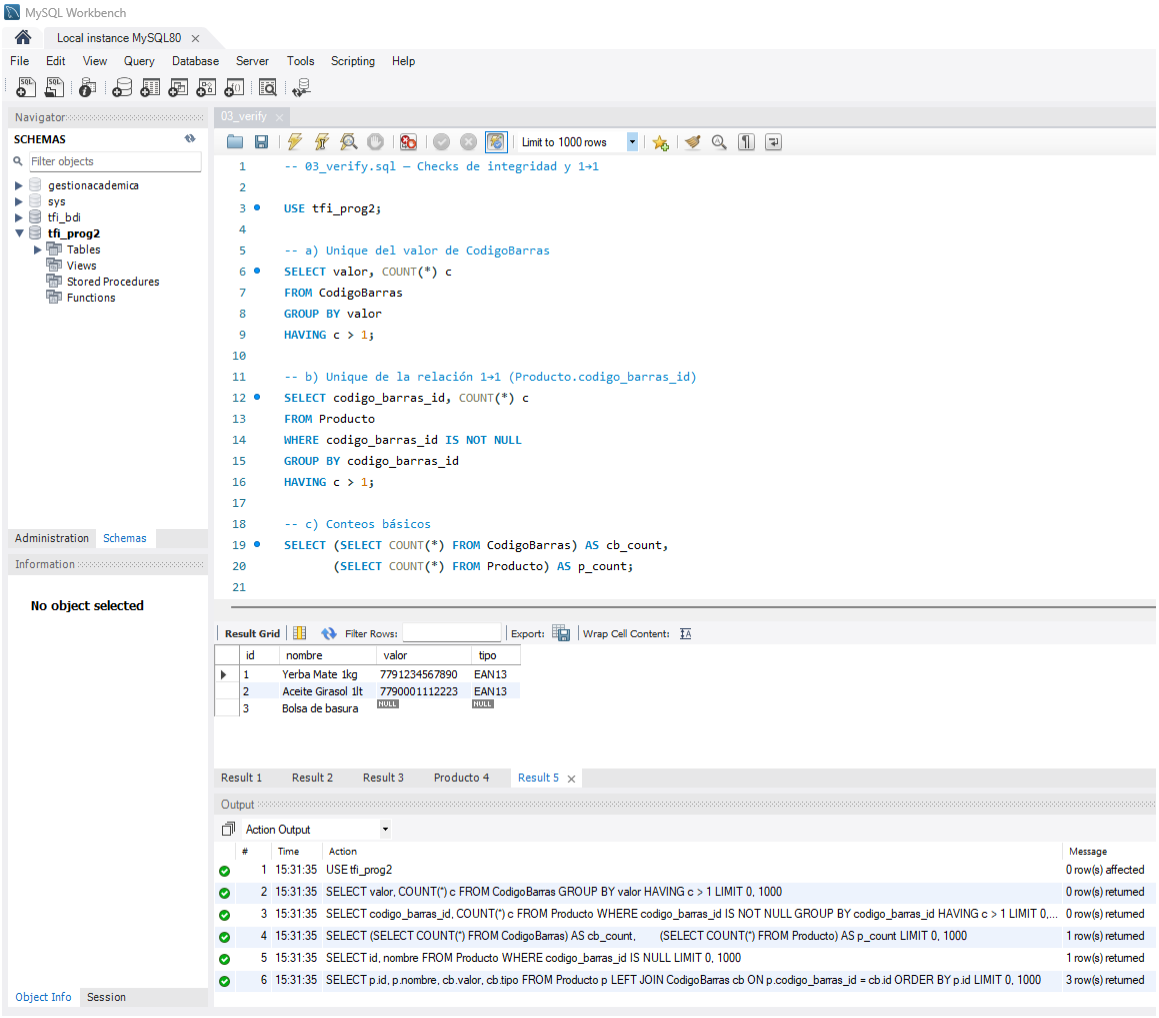
Capturas:

```
--- exec:3.1.0:exec (default-cli) @ tfi-programacion2 ---

=== TFI Programación 2 — Producto → CodigoBarras (1→1) ===
1) Alta CodigoBarras
2) Alta Producto + CodigoBarras NUEVO (transacción)
3) Alta Producto usando CodigoBarras EXISTENTE (transacción)
4) Listar Productos
5) Buscar Producto por valor de CodigoBarras
6) Eliminar lógico Producto
7) Listar Codigos de Barras
8) Eliminar lógico CodigoBarras
9) Demostración de rollback (UNIQUE valor)
0) Salir
Opción: 4
Producto{id=1, nombre='Yerba Mate 1kg', marca='Taragui', precio=4900.0, unidadMedida='kg', eliminado=false, codigoBarras={1,7791234567890}}
Producto{id=2, nombre='Aceite Girasol 1lt', marca='Natura', precio=3500.0, unidadMedida='lt', eliminado=false, codigoBarras={2,7790001112223}}
Producto{id=3, nombre='Bolsa de basura', marca='Gen|érica', precio=1200.0, unidadMedida='pz', eliminado=false, codigoBarras=null}

=== TFI Programación 2 — Producto → CodigoBarras (1→1) ===
1) Alta CodigoBarras
2) Alta Producto + CodigoBarras NUEVO (transacción)
3) Alta Producto usando CodigoBarras EXISTENTE (transacción)
4) Listar Productos
5) Buscar Producto por valor de CodigoBarras
6) Eliminar lógico Producto
7) Listar Codigos de Barras
8) Eliminar lógico CodigoBarras
9) Demostración de rollback (UNIQUE valor)
0) Salir
Opción: 7
CodigoBarras{id=1, valor='7791234567890', tipo='EAN13', eliminado=false}
CodigoBarras{id=2, valor='7790001112223', tipo='EAN13', eliminado=false}
CodigoBarras{id=3, valor='1234567890123', tipo='EAN13', eliminado=false}
```

```
=== TFI Programación 2 — Producto → CodigoBarras (1→1) ===
1) Alta CodigoBarras
2) Alta Producto + CodigoBarras NUEVO (transacción)
3) Alta Producto usando CodigoBarras EXISTENTE (transacción)
4) Listar Productos
5) Buscar Producto por valor de CodigoBarras
6) Eliminar lógico Producto
7) Listar Codigos de Barras
8) Eliminar lógico CodigoBarras
9) Demostración de rollback (UNIQUE valor)
0) Salir
Opción: 1
Valor (EAN/UPC): 7791234567890
Tipo [EAN13|UPC|EAN8] (default EAN13): EAN13
Duplicate entry '7791234567890' for key 'codigobarras.uq_codigobarras_valor'
```



---

8) Conclusiones y mejoras futuras

Conclusiones:

- Se implementó correctamente la relación **1→1 unidireccional** con **unicidades** efectivas en la BD.
- El patrón **DAO + Service** permitió separar persistencia de reglas de negocio y **manejar transacciones** de forma clara.
- El **AppMenu** facilita demostrar **CRUD**, **búsqueda por valor** y **rollback** con un flujo reproducible.

Mejoras futuras:

- Validaciones adicionales (por ejemplo, catálogo de tipos de código ampliado).

- **Logging** estructurado y auditoría (timestamps, usuario).
- Tests automatizados (JUnit) y **CI** básico.
- Dockerización de la app y la BD para facilitar la puesta en marcha.

---

9) Fuentes y herramientas utilizadas

- **Lenguaje y tecnologías:** Java 21, JDBC, MySQL 8, Maven.
- **Diseño y documentación:** Draw.io (UML), Markdown/Word.
- **Control y demo:** AppMenu (CLI), scripts SQL (01_schema.sql, 02_seed.sql, 03_verify.sql), run_all.sh / run_all.bat.
- **Asistencia de IA:** ChatGPT (GPT-5 Thinking) para apoyo en estructura del proyecto, checklist de pruebas, redacción del informe y guion del video.