

## Programación 2

# TRABAJO PRÁCTICO INTEGRADOR

### RELACIÓN: PRODUCTO → CÓDIGO DE BARRAS

**Coordinador:** Carlos Martinez

**Profesores:** Ariel Enferrel | Cinthia Rigoni | Alberto Cortez

**Tutor:** Neyén Bianchi

**Grupo: 117 – Integrantes:**

- Gaston Alberto Cejas – [gaston.cejas@gmail.com](mailto:gaston.cejas@gmail.com) - [GsuiteTdf/Programacion\\_TPI](https://GsuiteTdf/Programacion_TPI)
- Hernan Cóceres – [hernangc1998@gmail.com](mailto:hernangc1998@gmail.com) - [HernanCoceres/Programacion\\_TPI](https://HernanCoceres/Programacion_TPI)
- Claudio Rodriguez – [rodriguezop8@gmail.com](mailto:rodriguezop8@gmail.com) - [ClauRodriguez/Programacion\\_TPI](https://ClauRodriguez/Programacion_TPI)
- Hernan Bula – [hernanbula@gmail.com](mailto:hernanbula@gmail.com) - [hernanbula/Programacion\\_TPI](https://hernanbula/Programacion_TPI)



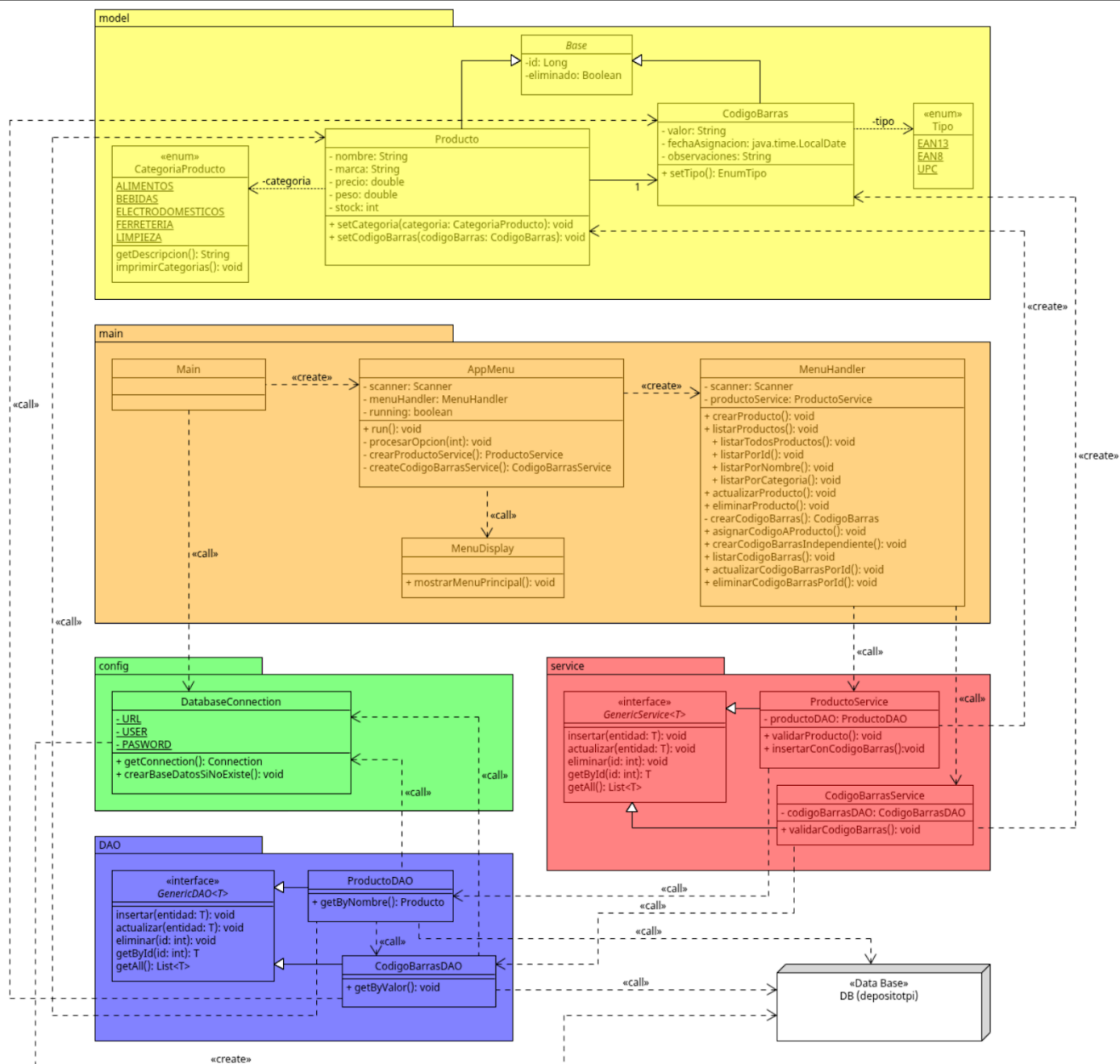
**LINK al video:** [click aqui \(https://youtu.be/QzxX1T7QYRg\)](https://youtu.be/QzxX1T7QYRg)

**LINK al repositorio Principal:** [CLICK AQUÍ](#)

*(Incluye además README.md, HISTORIAS\_DE\_USUARIO.md, javadocs)*

## DESARROLLO

### 1.- Etapa 1 – Decisiones de Diseño y Arquitectura



LINK al UML: [click aqui](#)

## 1.2. Arquitectura por Capas

Implementamos una arquitectura en 4 capas bien definidas mas una capa config como auxiliar:

**config/**: Gestión centralizada de la conexión a base de datos

```
java
public class DatabaseConnection {
    public static Connection getConnection() throws SQLException {
        // Implementación con parámetros externos
    }
    public static void inicializarBaseDatos() {
        // Creación automática de BD y tablas
    }
}
```

**model/**: Clases de dominio con herencia para campos comunes

```
java
public abstract class Base {
    protected Long id;
    protected Boolean eliminado;
    // getters, setters, constructores
}
```

```
public class Producto extends Base {
    private String nombre;
    private String marca;
    private String categoria;
    private Double precio;
    private Double peso;
    private Integer stock;
    private CodigoBarras codigoBarras; // Relación 1→1
}
```

**dao/**: Patrón DAO con interfaz genérica y implementaciones JDBC

```
java
public interface GenericDAO<T> {
    void insertar(T entidad, Connection conn) throws Exception;
    T leer(Long id, Connection conn) throws Exception;
    List<T> leerTodos(Connection conn) throws Exception;
    void actualizar(T entidad, Connection conn) throws Exception;
    void eliminar(Long id, Connection conn) throws Exception;
}
```

**service/**: Lógica de negocio y transacciones

```
java
public class ProductoService {
    public void insertarConCodigoBarras(Producto producto, CodigoBarras codigo) {
        // Transacción atómica
    }
}
```

main/: Interfaz de usuario y menú de consola

## 2.2. Decisiones Clave de Implementación

### Relación 1→1 Unidireccional:

- Optamos por FK única en tabla Producto hacia CodigoBarras
- Implementamos restricción UNIQUE para garantizar la cardinalidad
- ON DELETE SET NULL para preservar integridad referencial

### Transacciones:

- Todas las operaciones compuestas se ejecutan dentro de transacciones
- Control manual de autoCommit para garantizar atomicidad
- Rollback automático ante cualquier excepción

### Baja Lógica:

- Campo eliminado en clase base para soft delete
- Filtrado automático en consultas de listado
- Preservación de integridad referencial

## 2. Consultas al Equipo Docente

### Consulta 1 - 13/11/2025: Hernan Bula en el Foro de dudas y consultas

"Hola!

*Escribimos para consultar bien como es este tema del rollback, ya que no entendemos bien como hay que desarrollarlo en el TPI. Nosotros elegimos Producto y Código de Barras.*

*En cual de las capas iría esto? cómo se hace? Cómo se simula un fallo?*

*Estas consultas son porque en los criterios de evaluación está muy presente esto del rollback, pero tenemos muchas dudas de como solucionarlo y, sobre todo como hacer "muestra rollback con fallo simulado. → 30 pts"*

*Esperamos su respuesta.*

*muchas gracias?"*

**Respuesta del docente:** Ver [haciendo Click aquí](#)

### Consulta 2 - 18/10/2025: Vía – Clases de consulta

*"¿Deben los DAOs manejar las transacciones o esta responsabilidad es exclusiva de la capa Service?"*

### Respuesta del docente:

*"Los DAOs deben aceptar conexiones externas pero NO manejar transacciones. La capa Service debe orquestar las transacciones que involucren múltiples DAOs."*

## 3. Transacciones y Pruebas

### 3.1. Implementación de Transacciones

java

```
public void insertarConCodigoBarras(Producto producto, CodigoBarras codigo) throws Exception {
    Connection conn = null;
    try {
        conn = DatabaseConnection.getConnection();
        conn.setAutoCommit(false);

        // 1. Insertar código de barras
        codigoBarrasDAO.insertar(codigo, conn);

        // 2. Asociar producto con código
        producto.setCodigoBarras(codigo);
        productoDAO.insertar(producto, conn);

        conn.commit();
    } catch (Exception e) {
        if (conn != null) conn.rollback();
        throw e;
    } finally {
        if (conn != null) conn.setAutoCommit(true);
    }
}
```

## 3.2. Escenarios de Prueba

### Prueba 1 - Transacción Exitosa:

- Crear código de barras + crear producto asociado
- Verificar que ambas entidades persisten correctamente
- Confirmar relación en base de datos

### Prueba 2 - Rollback por Error:

- Intentar crear producto con código de barras duplicado
- Verificar que ninguna de las entidades se persiste
- Confirmar mensaje de error apropiado

### Prueba 3 - Validaciones de Negocio:

- Producto sin nombre → debe fallar
- Precio negativo → debe fallar
- Código de barras con valor duplicado → debe fallar

## 4. Herramientas y Fuentes Utilizadas

### 4.1. Stack Tecnológico

#### Desarrollo:

- Java JDK 21

- Apache NetBeans 20
- Git & GitHub

**Base de Datos:**

- MariaDB 10.6
- MySQL Connector/J 8.0

**Documentación:**

- UML Letino
- Microsoft Word para informe
- OBS Studio para grabación de video

**4.2. Fuentes de Consulta**

1. Documentación Oficial Oracle Java
2. MySQL 8.0 Reference Manual
3. Patrones de Diseño DAO - Martin Fowler
4. Documentación del TPI - UTN TUPaD

## 5. Uso Pedagógico de Inteligencia Artificial

### 5.1. Enfoque y Metodología

Implementamos la IA como herramienta de tutoría pedagógica, siguiendo los lineamientos institucionales. El proceso se caracterizó por:

- **Prompts específicos** y contextualizados
- **Validación crítica** de todas las respuestas
- **Iteración progresiva** en la resolución de problemas
- **Documentación completa** del proceso de aprendizaje

### 5.2. Interacciones con IA - Evidencias de Proceso

**Interacción 1 - 10/10/2025**

**Alumno:** Hernan Cóceres

**Capa:** Arquitectura general

**Prompt:** "Estoy comenzando el TPI de Programación 2 que requiere una relación 1→1 unidireccional en Java con MySQL. ¿Podrías explicarme los patrones arquitectónicos recomendados para este tipo de sistemas?"

**Respuesta IA:** "Para sistemas con persistencia relacional, se recomienda arquitectura en capas: capa de entidades (modelo), DAO (acceso a datos), Service (lógica de negocio) y presentación. El patrón DAO abstracto los datos, mientras Service maneja transacciones. ¿Qué dominio específico estás considerando?"

**Impacto:** Esta interacción nos ayudó a definir la estructura base del proyecto.

**Interacción 2 - 15/10/2025**

**Alumno:** Claudio Rodriguez

**Capa:** Entities

**Prompt:** "Necesito implementar dos clases Java para Producto y CódigoBarras con relación 1→1 unidireccional. ¿Cómo estructuro las clases para que sea fácil de mantener y cumpla con los requisitos de baja lógica?"

**Respuesta IA:** "Te sugiero una clase base abstracta con id y eliminado, que Producto y CodigoBarras extiendan. Producto debe tener referencia a CodigoBarras, pero no al revés. Implementa constructores, getters/setters y toString(). ¿Quieres que profundice en algún aspecto específico?"

**Impacto:** Definimos la herencia Base que simplificó la implementación de campos comunes.

**Interacción 3 - 20/10/2025**

**Alumno:** Hernan E. Bula

**Capa:** Base de datos

**Prompt:** "Para la relación 1→1 entre Producto y CodigoBarras en MySQL, ¿es mejor usar FK única en Producto o PK compartida? Necesito entender las ventajas de cada enfoque."

**Respuesta IA:** "FK única en Producto es más flexible - permite que un CodigoBarras exista sin Producto. PK compartida es más estricta pero puede complicar inserciones. Para tu caso, recomiendo FK única con restricción UNIQUE. ¿Qué opción se alinea mejor con tu dominio?"

**Impacto:** Optamos por FK única tras validar con el docente.

**Interacción 4 - 25/10/2025**

**Alumno:** Gaston A. Cejas

**Capa:** DAO

**Prompt:** "Estoy implementando el GenericDAO con JDBC. ¿Cómo diseño los métodos para que acepten conexión externa y puedan participar en transacciones managed por la capa Service?"

**Respuesta IA:** "Los métodos DAO deben recibir Connection como parámetro opcional. Si es null, crean su propia conexión; si no, usan la proporcionada. Esto permite transacciones multi-DAO. ¿Necesitas ayuda con la implementación concreta de algún método?"

**Impacto:** Implementamos el patrón de conexión externa exitosamente.

**Interacción 5 - 02/11/2025**

**Alumno:** Hernan Cóceres

**Capa:** Service



**Prompt:** "Debo implementar transacciones con commit/rollback en ProductService. ¿Podrías guiarme en la estructura try-catch-finally para garantizar atomicidad sin dejar conexiones abiertas?"

**Respuesta IA:** "Usa try-with-resources o asegúrate de cerrar conexiones en finally. Estructura básica: `setAutoCommit(false), try{operaciones; commit();} catch{rollback();} finally{setAutoCommit(true);}`. ¿Tienes dudas sobre el manejo de excepciones específicas?"

**Impacto:** Desarrollamos transacciones robustas con manejo seguro de recursos.

#### Interacción 6 - 07/11/2025

**Alumno:** Claudio Rodriguez

**Capa:** Validaciones

**Prompt:** "Necesito implementar validaciones de negocio en ProductService: nombre obligatorio, precio  $\geq 0$ , etc. ¿Debo hacerlas antes de la transacción o dentro de ella? ¿Cómo estructuro los mensajes de error?"

**Respuesta IA:** "Valida antes de iniciar la transacción para no consumir recursos innecesariamente. Usa excepciones específicas con mensajes claros para el usuario. Podrías crear una excepción `ValidacionException`. ¿Qué campos específicos necesitas validar?"

**Impacto:** Implementamos validaciones previas con mensajes descriptivos.

#### Interacción 7 - 10/11/2025

**Alumno:** Hernan E. Bula

**Capa:** Menú de consola

**Prompt:** "El menú de consola debe ser robusto ante entradas inválidas. ¿Cómo implemento un sistema de reintentos y validación de tipos sin que el programa falle?"

**Respuesta IA:** "Usa `Scanner.hasNextXXX()` para validar antes de leer. Implementa un bucle while que persista hasta entrada válida. Considera un menú recursivo con opción salir. ¿Quieres un ejemplo específico para validación numérica?"

**Impacto:** Creamos un menú resistente a errores de usuario.

#### Interacción 8 - 12/11/2025

**Alumno:** Gaston A. Cejas

**Capa:** Pruebas

**Prompt:** "Necesito probar el rollback de transacciones cuando hay error en la inserción de CodigoBarras. ¿Cómo simulo elegantemente un error para probar este escenario?"



**Respuesta IA:** "Puedes insertar deliberadamente un Código de Barras con valor duplicado, violando la restricción UNIQUE. O crear un método de prueba que lance excepción después de la primera inserción. ¿Buscas pruebas automatizadas o manuales?"

**Impacto:** Implementamos pruebas de rollback con datos duplicados.

#### Interacción 9 - 14/11/2025

**Alumno:** Hernan Cóceres

**Capa:** SQL y rendimiento

**Prompt:** "Nuestras tablas tienen índices en las FK y campos únicos. ¿Qué otros índices recomiendas para mejorar rendimiento en búsquedas por nombre de producto y categoría?"

**Respuesta IA:** "Crea índices compuestos en (categoria, nombre) si buscas por ambos campos, o índices separados si buscas individualmente. Considera también índice en eliminado para las queries que filtran por este campo. ¿Qué tipos de búsqueda son más frecuentes en tu sistema?"

**Impacto:** Optimizamos índices para las búsquedas más comunes.

#### Interacción 10 - 15/11/2025

**Alumno:** Claudio Rodriguez

**Capa:** Documentación

**Prompt:** "Estamos preparando la documentación final. ¿Qué elementos clave debemos incluir en el README.md para que los evaluadores puedan compilar y ejecutar fácilmente nuestro proyecto?"

**Respuesta IA:** "Incluye: requisitos exactos (Java version, BD), pasos para crear la BD, configuración de conexión, cómo compilar y ejecutar, credenciales de prueba, y troubleshooting común. Un ejemplo de flujo de uso ayuda mucho. ¿Tienes alguna particularidad en tu setup?"

**Impacto:** Elaboramos documentación completa y autocontenida.

### 5.3. Reflexión sobre el Uso de IA

El uso de IA generativa demostró ser una herramienta valiosa cuando se emplea con criterio pedagógico. Las principales ventajas observadas fueron:

- **Reducción de tiempo** en investigación de conceptos arquitectónicos
- **Ampliación de perspectivas** sobre diferentes enfoques de solución
- **Asistencia inmediata** en problemas de implementación específicos
- **Refuerzo de aprendizaje** a través de explicaciones alternativas

Sin embargo, mantuvimos una postura crítica:

- **Validamos sistemáticamente** todas las sugerencias con documentación oficial
- **Probamos exhaustivamente** cada solución propuesta antes de implementarla

- **Priorizamos el entendimiento** sobre la implementación ciega
- **Consultamos al docente** en decisiones arquitectónicas clave

Este enfoque nos permitió aprovechar las ventajas de la IA mientras desarrollábamos nuestras propias capacidades de análisis y solución de problemas.

## 6. Conclusiones y Aprendizajes

### 6.1. Logros del Proyecto

- ✓ Implementación exitosa de arquitectura en capas
- ✓ Relación 1→1 unidireccional funcional
- ✓ Sistema transaccional robusto con commit/rollback
- ✓ CRUD completo con validaciones de negocio
- ✓ Interfaz de usuario intuitiva y resistente a errores
- ✓ Documentación completa y profesional

### 6.2. Aprendizajes Técnicos

1. **Arquitectura de Software:** Aprendimos a diseñar sistemas modulares y mantenibles
2. **Patrones de Diseño:** Comprendimos la aplicación práctica de DAO y Service Layer
3. **Transacciones de BD:** Dominamos el manejo de atomicidad y consistencia
4. **Calidad de Código:** Implementamos prácticas de código limpio y manejo de excepciones

### 6.3. Mejoras Futuras

- Implementar interfaz gráfica de usuario (GUI)
- Agregar autenticación y autorización de usuarios
- Desarrollar API REST para integración con otros sistemas
- Implementar logging profesional y métricas de rendimiento
- Agregar pruebas unitarias automatizadas

## Conclusión Final

El desarrollo de este Trabajo Práctico Integrador nos permitió consolidar los conocimientos adquiridos en Programación II, demostrando capacidad para diseñar, implementar y documentar un sistema software completo.

El uso responsable de IA como herramienta de apoyo potenció nuestro aprendizaje autónomo, preparándonos para el uso profesional de estas tecnologías en el mercado laboral.