TRABAJO PRÁCTICO INTEGRADOR (TPI)

Programación II – UTN – Tecnicatura Universitaria en Programación a Distancia

Comisión 8 – Año 2025
Grupo 212
Sistema Pedido → Envío (Relación 1 → 1)

Integrantes:

- **Stefan Dios Mayarin**
- **Mathias Flor**
- **Joaquín Villaruel**
- **Ale Farfán**

Docente: Cinthia Rigoni
Tutor: Jerónimo Felipe Cortez
Fecha de entrega: 20/11/2025

1. Introducción

El presente Trabajo Práctico Integrador (TPI) se desarrolla en el marco de la asignatura Programación II, integrando conceptos fundamentales de programación orientada a objetos, arquitectura por capas, acceso a datos con JDBC, manejo de excepciones y transacciones.
El proyecto continúa la línea del TFI presentado previamente en la materia Bases de Datos I, en el cual se abordó un dominio más amplio de ventas. En esta oportunidad, el objetivo es implementar un sistema reducido y focalizado que modele la relación Pedido → Envío, cumpliendo estrictamente con los requerimientos establecidos por la cátedra.

El sistema propone gestionar dos entidades centrales —Pedido y Envío— en una relación 1 a 1 unidireccional, donde un pedido posee un único envío asociado. La consigna exige implementar un CRUD completo, baja lógica, validaciones, integridad relacional garantizada desde la base de datos y operaciones transaccionales.

El TPI también demanda la producción de documentación técnica, UML, scripts SQL reproducibles, un video demostrativo y un repositorio público en GitHub.

2. Integrantes y roles del equipo

A continuación, se detallan los roles y responsabilidades definidos por el grupo:

Integrante	Rol Asignado	Responsabilidades principales
Stefan Dios Mayarin	**Coordinador / QA / Menú y Pruebas**	**Coordinación general, ejecución de pruebas funcionales, armado del menú en consola, validación de casos de uso y funcionamiento.**

Mathias Flor	**Modelo de Dominio y UML**	**Implementación de entidades, enums, asociaciones y elaboración del diagrama UML en PlantUML.**
Joaquín Villaruel	**Base de Datos y Scripts SQL**	**Diseño de tablas, definición de PK y FK, garantía del 1→1 mediante UNIQUE, armado de scripts SQL y pruebas en MySQL.**
Ale Farfán	**DAO, JDBC y Servicios con Transacciones**	**Implementación de la capa DAO, lógica de negocio en Services, manejo de commit/rollback y validaciones.**

Esta división permitió desarrollar el proyecto en paralelo, manteniendo coherencia y cumplimiento del patrón arquitectónico solicitado.

3. Descripción del dominio: Pedido → Envío

El sistema administra dos entidades principales:

3.1. Pedido (A)

Representa una orden generada por un cliente. Incluye:

- **número de pedido**
- **fecha**
- **nombre del cliente**
- **total**
- **estado**
- **baja lógica (`eliminado`)**
- **envío asociado (opcional según método utilizado)**

3.2. Envío (B)

Contiene información logística vinculada al pedido:

- **código de tracking**
- **empresa transportista**
- **tipo de envío (estándar/exprés)**
- **costos**
- **fecha de despacho y fecha estimada**
- **estado del envío**
- **referencia al pedido**

3.3. Relación A → B (1→1 unidireccional)

La relación se implementa según los criterios de Programación II:

- **Un Pedido puede tener un único Envío.**
- **La relación es unidireccional desde Pedido hacia Envío.**

- En la base de datos se garantiza utilizando una FK con restricción UNIQUE en `envio.pedido_id`.

Esta decisión promueve integridad referencial, evita envíos duplicados y cumple estrictamente con el requerimiento de cardinalidad.

4. Requerimientos funcionales

El sistema implementa las siguientes operaciones:

4.1. Para Pedido

- Crear pedido
- Actualizar datos
- Buscar por ID
- Buscar por número (valor único)
- Listar todos
- Baja lógica

4.2. Para Envío

- Crear envío asociado a un pedido
- Actualizar envío
- Buscar por ID
- Baja lógica

4.3. Transacciones

Las operaciones críticas, como crear Pedido + Envío, se gestionan mediante transacciones:

- `setAutoCommit(false)`
- ejecución de ambos INSERT
- commit si ambas operaciones son exitosas
- rollback en caso de error

4.4. Persistencia

Toda la interacción con la base se realiza mediante:

- JDBC
- PreparedStatement
- ResultSet
- manejo de excepciones personalizadas (`ServiceException`)

5. Arquitectura del sistema

El proyecto sigue el patrón de arquitectura por capas, con separación completa de responsabilidades.

src/

├─ config/ → Conexión JDBC (DatabaseConnection)

├─ entities/ → Pedido, Envio, enums y clases de dominio

├─ dao/ → Interfaces y DAO JDBC (PedidoDao, EnvioDao)

├─ service/ → Reglas de negocio, validaciones y transacciones

└─ main/ → AppMenu y punto de entrada (Main)

Justificación del diseño

- **Entities modela el dominio de forma independiente.**
- **DAO encapsula operaciones de persistencia.**
- **Service desacopla lógica de negocio y transacciones del acceso a datos.**
- **Main/AppMenu ofrece la interfaz de uso en consola.**

Este diseño cumple con las expectativas de Programación II y facilita mantenibilidad.

6. Modelo entidad–relación y UML

6.1. Tablas del sistema

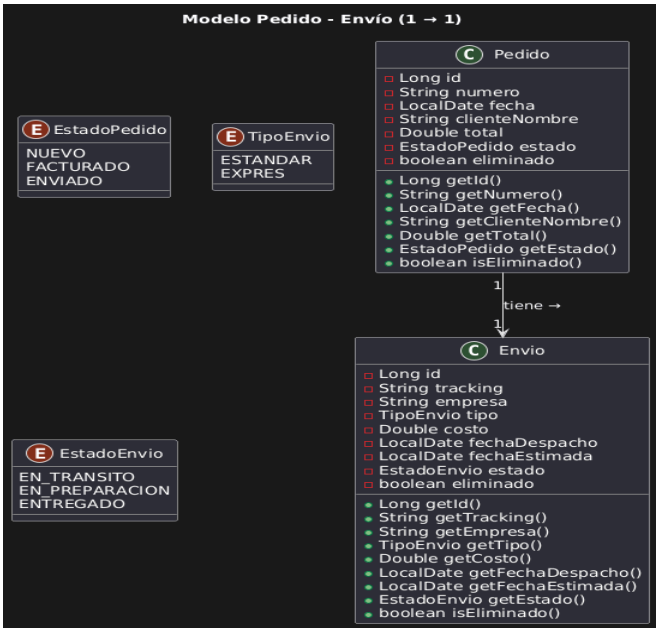
- **pedido (PK: id)**
- **envio (PK: id, FK: pedido_id UNIQUE)****

6.2. Relación

`pedido (1) - envio (1)`

La integridad se garantiza desde la tabla envio, donde `pedido_id` tiene una restricción UNIQUE.

6.3. Diagrama UML



7. Base de datos (MySQL)

El diseño fue implementado en dos scripts:

7.1. Script 01 – Creación de estructura

Incluye:

- **creación de schema**
- **creación de tabla pedido**
- **creación de tabla envio**
- **FK con UNIQUE**
- **atributos y tipos según consigna**

01_create_database.sql

```
CREATE DATABASE IF NOT EXISTS tp_p2_pedido_envio
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
USE tp_p2_pedido_envio;
-- Tabla Pedido (A)
CREATE TABLE pedido (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    eliminado TINYINT(1) NOT NULL DEFAULT 0,
    numero VARCHAR(20) NOT NULL UNIQUE,
    fecha DATE NOT NULL,
    cliente_nombre VARCHAR(120) NOT NULL,
    total DECIMAL(12,2) NOT NULL,
    estado ENUM('NUEVO', 'FACTURADO', 'ENVIADO') NOT NULL
);
-- Tabla Envio (B)
CREATE TABLE envio (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    eliminado TINYINT(1) NOT NULL DEFAULT 0,
    tracking VARCHAR(40) UNIQUE,
    empresa ENUM('ANDREANI', 'OCA', 'CORREO_ARG'),
    tipo ENUM('ESTANDAR', 'EXPRES'),
    costo DECIMAL(10,2),
    fecha_despacho DATE,
    fecha_estimada DATE,
    estado ENUM('EN_PREPARACION', 'EN_TRANSITO', 'ENTREGADO'),
```

pedido_id BIGINT UNIQUE,

CONSTRAINT fk_envio_pedido

FOREIGN KEY (pedido_id)

REFERENCES pedido(id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

7.2. Script 02 – Inserción de datos

Incluye:

- **inserción de pedidos de prueba**
- **inserción de envíos asociados**
- **validación de la FK UNIQUE**

02_insert_data.sql

USE tp_p2_pedido_envio;

INSERT INTO pedido (numero, fecha, cliente_nombre, total, estado)

VALUES

('PED-0001', '2025-11-01', 'Juan Pérez', 15000.00, 'NUEVO'),

('PED-0002', '2025-11-02', 'María Gómez', 24500.50, 'FACTURADO'),

('PED-0003', '2025-11-03', 'Carlos López', 8000.00, 'ENVIADO');

INSERT INTO envio (tracking, empresa, tipo, costo,

fecha_despacho, fecha_estimada,

estado, pedido_id)

VALUES

('TRK-AND-0001', 'ANDREANI', 'ESTANDAR', 1500.00, '2025-11-02', '2025-11-05', 'EN_TRANSITO', 1),

('TRK-OCA-0002', 'OCA', 'EXPRES', 2000.00, '2025-11-03', '2025-11-04', 'EN_PREPARACION', 2),

('TRK-COR-0003', 'CORREO_ARG', 'ESTANDAR', 1300.00, '2025-11-04', '2025-11-08', 'ENTREGADO', 3);

8. Implementación

8.1. DAO

Los DAO utilizan:

- `Connection` **externa (cumpliendo la consigna)**

- `PreparedStatement`
- captura controlada de excepciones
- mappeo de ResultSet a entidades

Clases principales:

- `PedidoDaoImpl`
- `EnvioDaoImpl`

8.2. Capa Service

Responsable de:

- validaciones
- orquestación de transacciones
- lógica de negocio
- atomicidad de operaciones

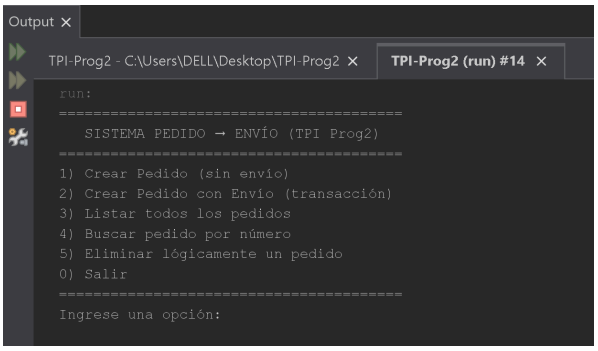
Incluye:

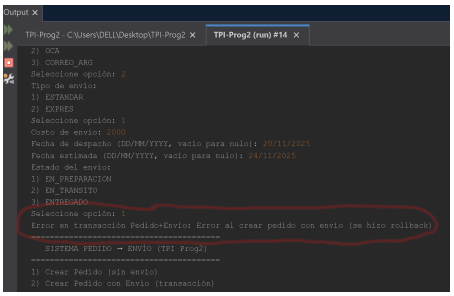
- `insertarConEnvioTx()` con commit/rollback
- Exceptions propias (`ServiceException`)

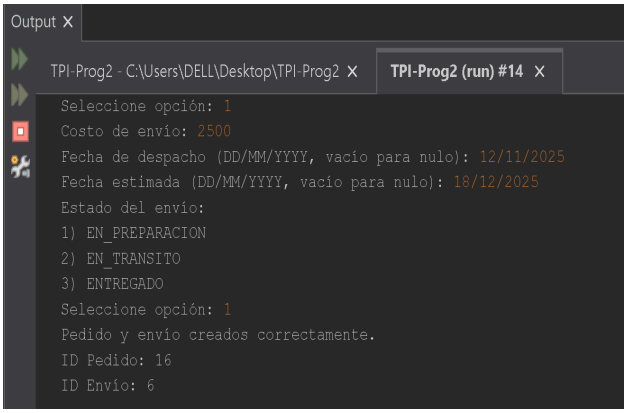
8.3. Interfaz AppMenu

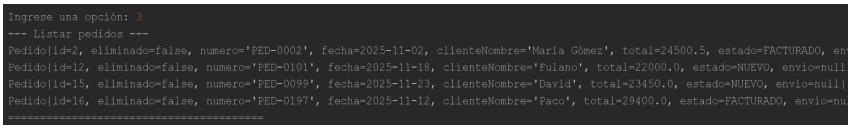
Gestiona operaciones en consola:

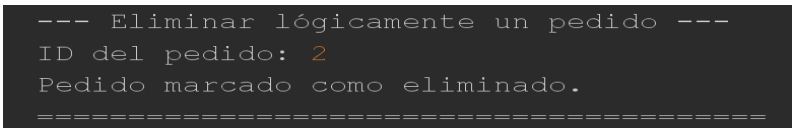
- Crear Pedido
- Crear Pedido con Envío
- Listar
- Buscar
- Eliminar lógicamente
- Salir

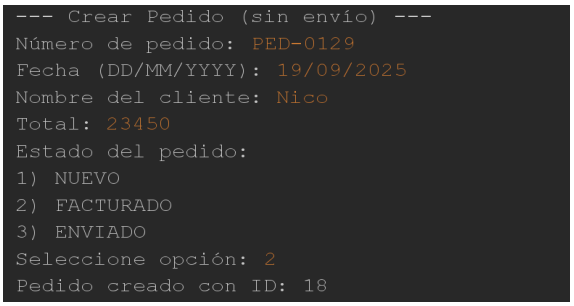












9. Pruebas realizadas

El grupo efectuó pruebas sobre:

✔ CRUD Pedido

✔ CRUD Envío

✔ Relación 1 → 1

✔ Baja lógica

✔ Transacciones exitosas y fallidas

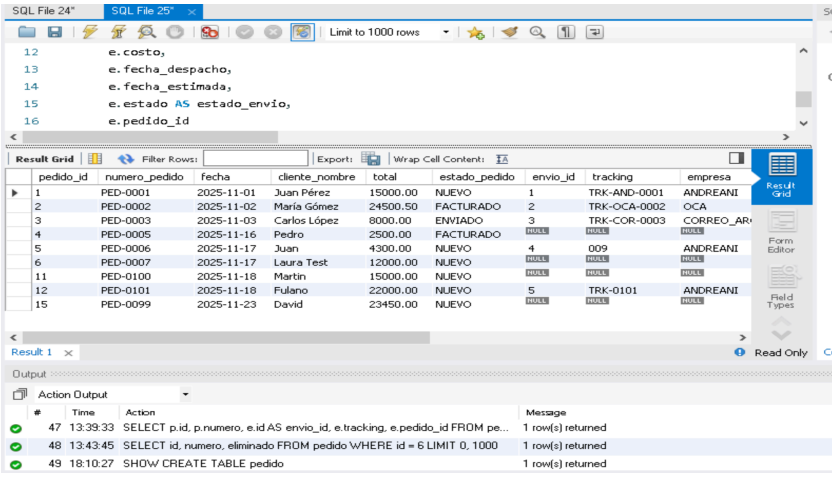
✔ Integridad referencial

✔ Errores controlados

Incluye capturas de:

- **ejecución del menú**

- mensajes de éxito
- rollback al forzar errores
- consultas SQL de verificación



10. Conclusiones

El desarrollo del TPI permitió:

- **Aplicar conceptos completos de Programación Orientada a Objetos.**
- **Implementar correctamente un modelo 1→1 con integridad garantizada.**
- **Utilizar JDBC de forma profesional.**
- **Comprender el patrón DAO/Service.**
- **Trabajar con transacciones reales.**
- **Integrar Programación II con Bases de Datos I.**
- **Organizarse en equipo mediante Git y GitHub.**

El sistema final cumple con todos los requerimientos funcionales, técnicos y documentales solicitados por la cátedra.

11. Declaración de uso de IA

El grupo utilizó herramientas de Inteligencia Artificial (ChatGPT) únicamente como apoyo pedagógico para:

- **aclarar dudas de sintaxis**
- **mejorar la documentación**
- **organizar la arquitectura**
- **identificar errores y alternativas de diseño**