Trabajo Final Integrador

Pedido-Envío

Alumnos:

Daniela Romero daniela.romero@tupad.utn.edu.ar

Esteban Rivarola esteban.rivarola@tupad.utn.edu.ar

Agustin Rivarola agustin.rivarola@tupad.utn.edu.ar

Materia: Programación II

Profesor: Prof. Ariel Enferrel

Fecha de Entrega: 20/11/2025

Índice

Objetivos de Aprendizaje de este Trabajo Práctico:

Desarrollar una aplicación en Java que modele dos clases relacionadas mediante un asociación unidireccional 1 a 1 (la clase “A” referencia a la clase “B”), persistiendo datos en una base relacional mediante JDBC y el patrón DAO, con operaciones transaccionales (commit/rollback) y menú de consola para CRUD.

1-Integrantes y roles asignados

El grupo se integró inicialmente por cuatro estudiantes pero, habiendo uno de ellos abandonado la cursada, terminó teniendo 3. Los tres estudiantes estuvieron involucrados en todas las etapas de diseño del proyecto, pero a continuación se detalla en cuales se destacaron por sus aportes:

Daniela Romero	Gestión de Versiones (Git/GitHub):Liderazgo en el uso de Git, gestión de ramas (merge) y resolución de conflictos. Capa de Conexión (Config):Creación e implementación de la clase DatabaseConnection y el DBInitializer.Diseño y creación inicial del esquema (init.sql) y pruebas de conexión. Integridad y Consistencia de Datos: Implementación de la Eliminación Lógica (Soft Delete / eliminado = 1/0) en la base y DAOs. Debugging Crítico: Corrección de errores de conexión y código.
Agustin Rivarola	Capa de Negocio (Services): Implementación de los Servicios (PedidoServiceImpl, EnvioServiceImpl) que contienen la lógica de negocio, validaciones y uso de transacciones. Modelos (Entities): Creación de las clases modelo (Pedido, Envio, Base) y los Enums asociados (Estado, Empresa, TipoEnvio, etc.). Utilidades: Desarrollo de lógica auxiliar (uniquesGenerator) para generar códigos de pedido y tracking. Documentación/Presentación: Responsable de la generación final del material de presentación (diapositivas, PDF).
Esteban Rivarola	Modelado de Datos (UML): Construcción del diagrama de clases (UML) para definir la estructura del sistema. Capa de Acceso a Datos (DAO): Implementación de las clases DAO (PedidoDAO, EnvioDAO) con los métodos CRUD y lógica SQL específica. Incluyendo desarrollo de las queries SQL compleja. Interfaz de Usuario (Menús): Construcción de los menús de navegación (MenuDisplay, MenuHandler) y lógica de interacción básica.

2. Elección del Dominio y Justificación

Dominio Elegido: Pedido ⟶ Envío (Relación 1⟶1 Unidireccional)

- Clase A: Pedido
- Clase B: Envio

Relación: 1⟶1 (la clase Pedido contiene la referencia al objeto Envio).
El modelo cumple con la asociación obligatoria 1⟶1 donde cada pedido necesita un único registro de envío asociado para su trazabilidad logística

3. Diseño del Sistema

3.1. Decisiones de Diseño Clave

Relación 1⟶1 en la Base de Datos: Se implementa utilizando una clave foránea única en la tabla de Envios. Esto garantiza que un id_pedido sólo pueda aparecer una vez en la tabla Envios, cumpliendo con la relación 1⟶1.

Acción de Borrado: Se implementa un ON DELETE CASCADE en la FK (id_pedido) para garantizar la integridad referencial.

3.2. Diagrama UML de Clases

Se adjunta PDF con diagrama UML de Clases del proyecto. Para verlo haga click aquí.
https://github.com/Daniela-N-Romero/TFI-ProgamacionII/blob/main/TFIprogramacion2UML.png

4. Arquitectura por Capas

El proyecto está organizado en las siguientes capas, siguiendo el patrón DAO y la arquitectura por capas:

- Config/: Gestionar la conexión a la base de datos (DatabaseConnection.java , TransactionManager.java, DBInitializer, init.sql)

- DAO/ : Implementación del Patrón DAO. Responsables del acceso a datos usando JDBC y PreparedStatement. Los métodos aceptan una Connection externa para transacciones. (GenericDAO.java , PedidoDAO.java , EnvioDAO.java)

- Models/ : Clases de dominio. Contienen atributos, getters/setters y el campo eliminado (Base.java, Pedido.java, Envio.java, nums: (Estado.java, EstadoEnvio.java, TipoEnvio.java)

- Service/ : Capa de Negocio y Transacciones. Responsable de la lógica de negocio, validaciones y la orquestación de operaciones compuestas mediante

commit/rollback sobre una conexión compartida. (GenericService.java, PedidoServiceImpl.java, EnvioServiceImpl.java)

- Main/: Contiene el punto de entrada de la aplicación y la lógica de la Interfaz de Consola (UI). Es la capa de presentación que maneja la interacción con el usuario. (AppMenu.java, Main.java, MenuDisplay.java, MenuHandler.java, TestConexion.java)

- Utils/: Contiene clases de utilidad que realizan tareas específicas y reutilizables, no directamente relacionadas con las reglas de negocio principales, sino con la generación y manipulación de datos. (uniquesGenerator.java).

5. Persistencia: Base de Datos y Transacciones

5.1. Estructura de la Base de Datos

El diseño utiliza dos tablas principales: pedidos (que corresponde a la Clase A) y envíos (Clase B). Ambas tablas incluyen un campo id como clave primaria y el campo booleano eliminado

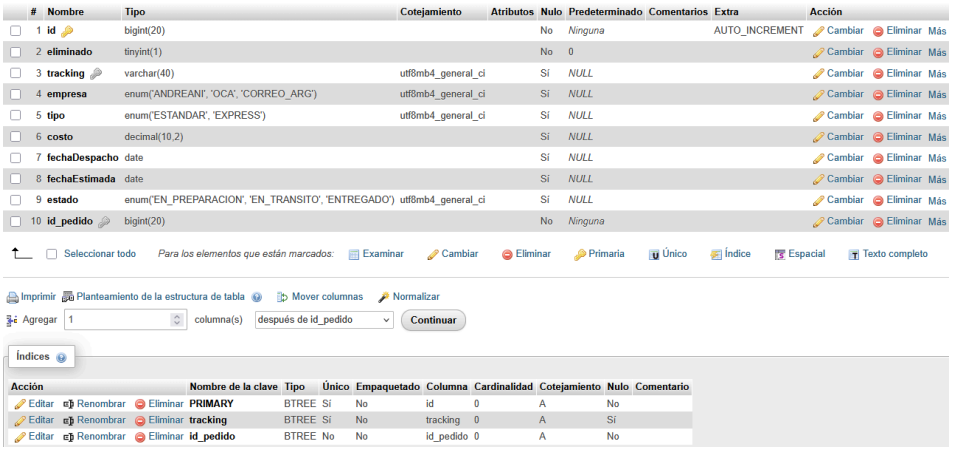
Para cumplir con la relación 1⟶1 unidireccional es la definición de la tabla envíos. Esta tabla contiene una Clave Foránea (id_pedido) que referencia a la tabla pedidos, y se le aplica la restricción UNIQUE a dicha clave foránea. Esto garantiza la integridad referencial y evita que un Pedido pueda tener más de un registro de Envio asociado.

- Herramienta: phpMyAdmin , XAMPP Control Panel

- Tablas: Esquema :

La base de datos (tfi_db) se compone de dos tablas, pedidos y envíos, que reflejan las clases de dominio A y B.

- Tabla Envio



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	bigint(20)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	eliminado	tinyint(1)			No	0			Cambiar Eliminar Más
3	tracking	varchar(40)	utf8mb4_general_ci		Sí	*NULL*			Cambiar Eliminar Más
4	empresa	enum('ANDREANI', 'OCA', 'CORREO_ARG')	utf8mb4_general_ci		Sí	*NULL*			Cambiar Eliminar Más
5	tipo	enum('ESTANDAR', 'EXPRESS')	utf8mb4_general_ci		Sí	*NULL*			Cambiar Eliminar Más
6	costo	decimal(10,2)			Sí	*NULL*			Cambiar Eliminar Más
7	fechaDespacho	date			Sí	*NULL*			Cambiar Eliminar Más
8	fechaEstimada	date			Sí	*NULL*			Cambiar Eliminar Más
9	estado	enum('EN_PREPARACION', 'EN_TRANSITO', 'ENTREGADO')	utf8mb4_general_ci		Sí	*NULL*			Cambiar Eliminar Más
10	id_pedido	bigint(20)			No	*Ninguna*			Cambiar Eliminar Más

Seleccionar todo *Para los elementos que están marcados:* Examinar Cambiar Eliminar Primaria Único Índice Espacial Texto completo

Imprimir Planteamiento de la estructura de tabla Mover columnas Normalizar

Agregar 1 columna(s) después de id_pedido Continuar

Índices

Acción	Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
Editar Renombrar Eliminar	PRIMARY	BTREE	Sí	No	id	0	A	No	
Editar Renombrar Eliminar	tracking	BTREE	Sí	No	tracking	0	A	Sí	
Editar Renombrar Eliminar	id_pedido	BTREE	No	No	id_pedido	0	A	No	

- Tabla Pedido

Estructura de tabla | Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	bigint(20)			No	*Ninguna*		AUTO_INCREME
2	eliminado	tinyint(1)			No	0		
3	tracking	varchar(40)	utf8mb4_general_ci		Sí	*NULL*		
4	empresa	enum('ANDREANI', 'OCA', 'CORREO_ARG')	utf8mb4_general_ci		Sí	*NULL*		
5	tipo	enum('ESTANDAR', 'EXPRESS')	utf8mb4_general_ci		Sí	*NULL*		
6	costo	decimal(10,2)			Sí	*NULL*		
7	fechaDespacho	date			Sí	*NULL*		
8	fechaEstimada	date			Sí	*NULL*		
9	estado	enum('EN_PREPARACION', 'EN_TRANSITO', 'ENTREGADO')	utf8mb4_general_ci		Sí	*NULL*		
10	id_pedido	bigint(20)			No	*Ninguna*		

Seleccionar todo *Para los elementos que están marcados:* Examinar Cambiar Eliminar Primaria

6. Validaciones y Reglas de Negocio

La capa Service es la responsable de las transacciones obligatorias.
La operación de crearPedido(pedido) es transaccional y se maneja en el PedidoServiceImpl
Otro punto fundamental es que actúa como la capa de negocio central y es la única que tiene acceso al TransactionManager. Sus responsabilidades principales son:

- Validación de Reglas de Negocio:

Antes de realizar cualquier persistencia, métodos privados como validatePedido() y validateEnvio() verifican la integridad de los datos como por ejemplo: campos obligatorios, valores positivos , formato.

- Control Transaccional:

Utiliza el TransactionManager inyectado para definir los límites de la operación. Se asegura que cada operación de escritura
- Inicia la transacción (txManager.startTransaction())
- Finaliza exitosamente con commit() si no hay errores.
- Finaliza con rollback() si ocurre alguna excepción, garantizando la atomicidad.

- Orquestacion del 1 → 1:

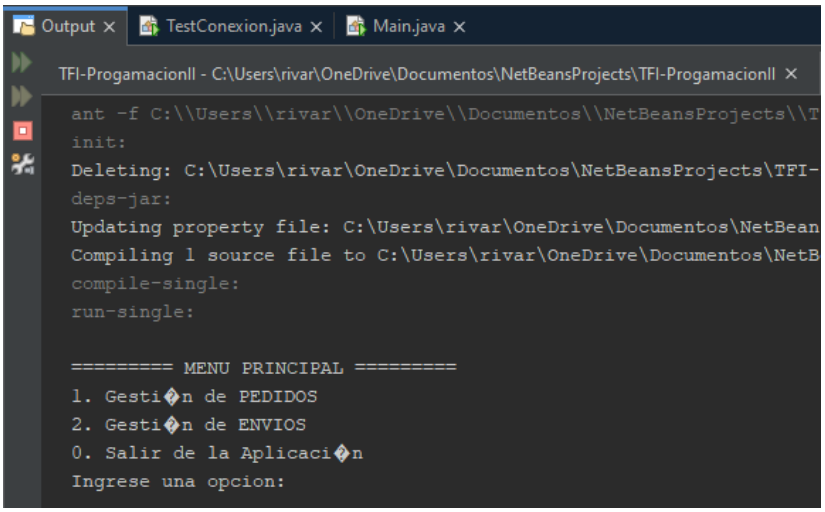
El PedidoServiceImpl.insertar(Pedido) coordina la creación del Pedido y del Envío en una única transacción:
- Inserta el Pedido para obtener su ID autogenerado
- Asigna este ID al objeto Envio (envio.setIdPedido(pedido.getId())).
- Inserta el Envío, garantizando la FK y la unicidad 1⟶1.

- Baja Lógica Compuesta:

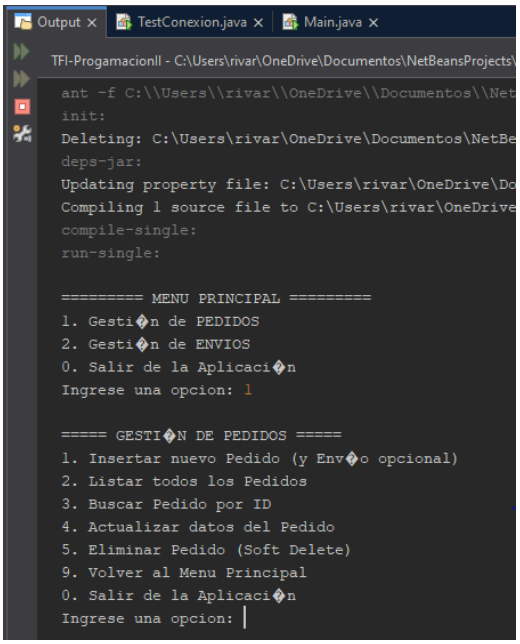
En la operación eliminar(id), el PedidoServiceImpl ejecuta la baja lógica sobre el Pedido y su Envío asociado (pedidoDAO.eliminarTx y envioDAO.eliminarTx) dentro del mismo bloque transaccional.

7. Pruebas Realizadas y Conclusiones

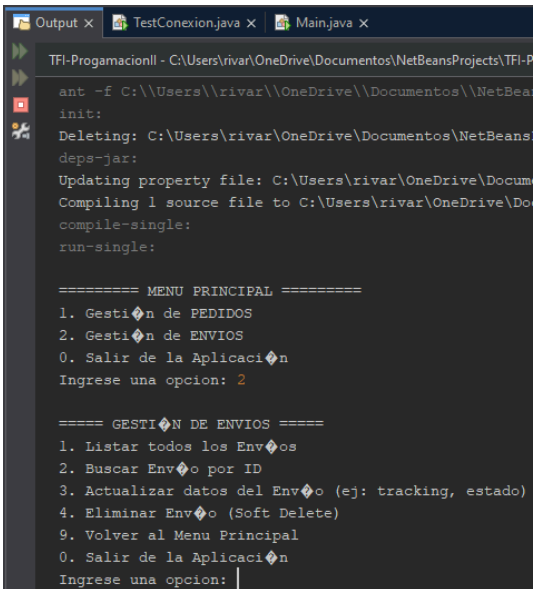
- Menú Principal:



- Menú Gestión Pedidos:



- Menú Gestión Envios:



- Consulta SQL

```
===== GESTI□N DE PEDIDOS =====
1. Insertar nuevo Pedido (y Env□o opcional)
2. Listar todos los Pedidos
3. Buscar Pedido por ID
4. Actualizar datos del Pedido
5. Eliminar Pedido (Soft Delete)
9. Volver al Menu Principal
0. Salir de la Aplicaci□n
Ingrese una opcion: 2
ID: 2, Numero de pedido: P20250002, Fecha: 2025-11-17, Nombre de cliente: Mar□a Rodr□guez, Total: 12500.00, Estado del pedido: FACTURADO
 Envio: Envio{ ID=3tracking=OC2025T010, costo=200.00, fechaDespacho=2025-11-19, fechaEstimada=2025-11-28, empresa=CORREO_ARG, estado=EN_PREPARACION, tipo=ESTANDAR, eliminado=false}
ID: 1, Numero de pedido: P20250001, Fecha: 2025-11-17, Nombre de cliente: Juan P□rez, Total: 4500.50, Estado del pedido: NUEVO
 Envio: Envio{ ID=2tracking=OC2025T011, costo=200.00, fechaDespacho=2025-11-19, fechaEstimada=2025-11-28, empresa=CORREO_ARG, estado=EN_PREPARACION, tipo=ESTANDAR, eliminado=false}
ID: 6, Numero de pedido: P20250006, Fecha: 2025-11-17, Nombre de cliente: Esteban Rivarola, Total: 5500.00, Estado del pedido: NUEVO
 (Retiro en Local)
ID: 7, Numero de pedido: P20250007, Fecha: 2025-11-17, Nombre de cliente: Horacio Quiroga, Total: 7500.00, Estado del pedido: NUEVO
 (Retiro en Local)
ID: 8, Numero de pedido: P20250008, Fecha: 2025-11-17, Nombre de cliente: Jean Claude, Total: 12000.00, Estado del pedido: NUEVO
 Envio: Envio{ ID=7tracking=TRK02488, costo=1000.00, fechaDespacho=2025-11-18, fechaEstimada=2025-11-22, empresa=CORREO_ARG, estado=EN_PREPARACION, tipo=ESTANDAR, eliminado=false}
ID: 9, Numero de pedido: P20250009, Fecha: 2025-11-17, Nombre de cliente: Mirtha Legrand, Total: 50.00, Estado del pedido: NUEVO
 (Retiro en Local)
ID: 3, Numero de pedido: P20250003, Fecha: 2025-11-16, Nombre de cliente: Carlos G□mez, Total: 890.99, Estado del pedido: ENVIADO
 Envio: Envio{ ID=4tracking=OC2025T003, costo=400.99, fechaDespacho=2025-11-19, fechaEstimada=2025-11-28, empresa=OCA, estado=EN_PREPARACION, tipo=ESTANDAR, eliminado=false}
ID: 4, Numero de pedido: P20250004, Fecha: 2025-11-15, Nombre de cliente: Ana Torres, Total: 32000.75, Estado del pedido: NUEVO
 Envio: Envio{ ID=5tracking=AR2025T004, costo=3500.00, fechaDespacho=2025-11-15, fechaEstimada=2025-11-18, empresa=ANDREANI, estado=EN_PREPARACION, tipo=EXPRESS, eliminado=false}
ID: 5, Numero de pedido: P20250005, Fecha: 2025-11-15, Nombre de cliente: Roberto Gomez Bolanios, Total: 2500.00, Estado del pedido: NUEVO
 Envio: Envio{ ID=6tracking=CA2025T005, costo=300.00, fechaDespacho=2025-11-18, fechaEstimada=2025-11-23, empresa=CORREO_ARG, estado=EN_TRANSITO, tipo=ESTANDAR, eliminado=false}
```

- Comparación con los datos de la Base de Datos:

	id	eliminado	numero_pedido	fecha	cliente_nombre	total	estado
Editar Copiar Borrar	1	0	P20250001	2025-11-17 10:30:00	Juan Pérez	4500.50	NUEVO
Editar Copiar Borrar	2	0	P20250002	2025-11-17 11:45:00	María Rodríguez	12500.00	FACTURADO
Editar Copiar Borrar	3	0	P20250003	2025-11-16 16:20:00	Carlos Gómez	890.99	ENVIADO
Editar Copiar Borrar	4	0	P20250004	2025-11-15 09:00:00	Ana Torres	32000.75	NUEVO
Editar Copiar Borrar	5	0	P20250005	2025-11-15 00:00:00	Roberto Gomez Bolanios	2500.00	NUEVO
Editar Copiar Borrar	6	0	P20250006	2025-11-17 00:00:00	Esteban Rivarola	5500.00	NUEVO
Editar Copiar Borrar	7	0	P20250007	2025-11-17 00:00:00	Horacio Quiroga	7500.00	NUEVO
Editar Copiar Borrar	8	0	P20250008	2025-11-17 00:00:00	Jean Claude	12000.00	NUEVO
Editar Copiar Borrar	9	0	P20250009	2025-11-17 00:00:00	Mirtha Legrand	50.00	NUEVO

- Conclusiones :

El desarrollo de este Trabajo Final Integrador nos permitió cumplir con los objetivos principales establecidos:

1. modelar una relación 1⟶1 unidireccional (Pedido ⟶ Envío)
2. implementar el Patrón DAO para la persistencia
3. Gestionar operaciones transaccionales (commit/rollback) usando JDBC sobre MySQL.

El desafío principal se nos presentó en la implementación de la arquitectura por capas (DAO/Service). Si bien inicialmente la separación de responsabilidades entre la persistencia (DAO) y la lógica de negocio/transacciones (Service) presentó una dificultad, el esfuerzo resultó en una comprensión profunda de cómo organizar el código para mejorar la reutilización, mantenimiento y testeo. Se constató que, a pesar de la dificultad inicial en la abstracción de este modelo, la organización del trabajo en capas es fundamental para proyectos futuros, facilitando la escalabilidad y la claridad del flujo de

datos.

Otra gran oportunidad de aprendizaje fue generada por la gestión de dependencias y el trabajo con Github. El proceso de desarrollo colaborativo reveló un desafío con la gestión del driver de la base de datos (MariaDB). Inicialmente, la configuración del driver basada en rutas de acceso locales causó conflictos de versiones y fallos de compilación en los entornos de los distintos miembros del equipo. La solución arquitectónica adoptada fue consolidar el driver JDBC en una carpeta /lib dentro del proyecto, asegurando que la dependencia fuera manejada de manera relativa al repositorio. Esto garantizó la portabilidad del código y eliminó los conflictos de ruta locales, estandarizando el entorno de desarrollo.

Además, un objetivo clave fue lograr que la aplicación pudiera ejecutarse "desde cero" sin intervención manual en la base de datos. Esto se logró con la clase DBInitializer. La conclusión fundamental en este punto fue la importancia de la idempotencia en el script SQL de inicialización (init.sql). Para evitar errores de duplicidad al correr la aplicación múltiples veces, fue necesario el uso de comandos condicionales: CREATE DATABASE IF NOT EXISTS, CREATE TABLE IF NOT EXISTS y, especialmente, INSERT IGNORE INTO para la carga de datos de prueba (Seed Data). Esto garantiza un arranque limpio y robusto en cualquier entorno.

Por último, proponemos que para los futuros grupos de estudiantes de la cátedra Programación 2 éste trabajo se realice de manera gradual, donde en cada unidad haya ejercitaciones o demostraciones que vayan aportando poco a poco a dicho trabajo. Creemos que darle más peso e importancia a la unidad de “Acceso a Base de Datos” hubiera facilitado la resolución del presente trabajo.

Enlace a vídeo explicativo

A continuación se adjunta enlace a video:

https://www.youtube.com/watch?v=FtIR3893st8

Enlace a repositorio en GitHub

https://github.com/Daniela-N-Romero/TFI-ProgramacionII