UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Final de Integración II

Sistema de Gestión de Pedidos y Envíos

Alumnos - Comision 4

Gaston Paschetta

Javier Ovelar

Bruno Maza

Karina Rodriguez

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional

Programación II

Dominio Elegido

Para el desarrollo del TFI se eligió la pareja **Pedido** ⇨ **Envío**, donde la clase Pedido contiene la referencia unidireccional 1 ⇨ 1 a la clase Envío.
Esta elección se justifica por la naturaleza atómica de la operación de creación, que requiere obligatoriamente el uso de transacciones en la capa de persistencia.
En un escenario de comercio electrónico, por ejemplo, cuando un cliente finaliza una compra se inicia una operación que no puede dividirse sin comprometer la integridad del sistema.
Esta operación requiere que dos acciones relacionadas y críticas ocurran de forma conjunta en la base de datos:

1. La creación del registro Envío.
2. La creación del registro Pedido, el cual debe incluir la referencia al Envío recién creado.

Estas dos operaciones de inserción deben tratarse como una unidad atómica de trabajo para evitar un posible estado de inconsistencia en la DB si por alguna razón falla la creación de alguno de los dos registros.
Para evitar esta inconsistencia, el sistema debe utilizar transacciones. Esto se logra en la capa Service, mediante la configuración del setAutoCommit(false) y la ejecución de commit() si ambas operaciones tienen éxito, o un rollback() si alguna de ellas falla.

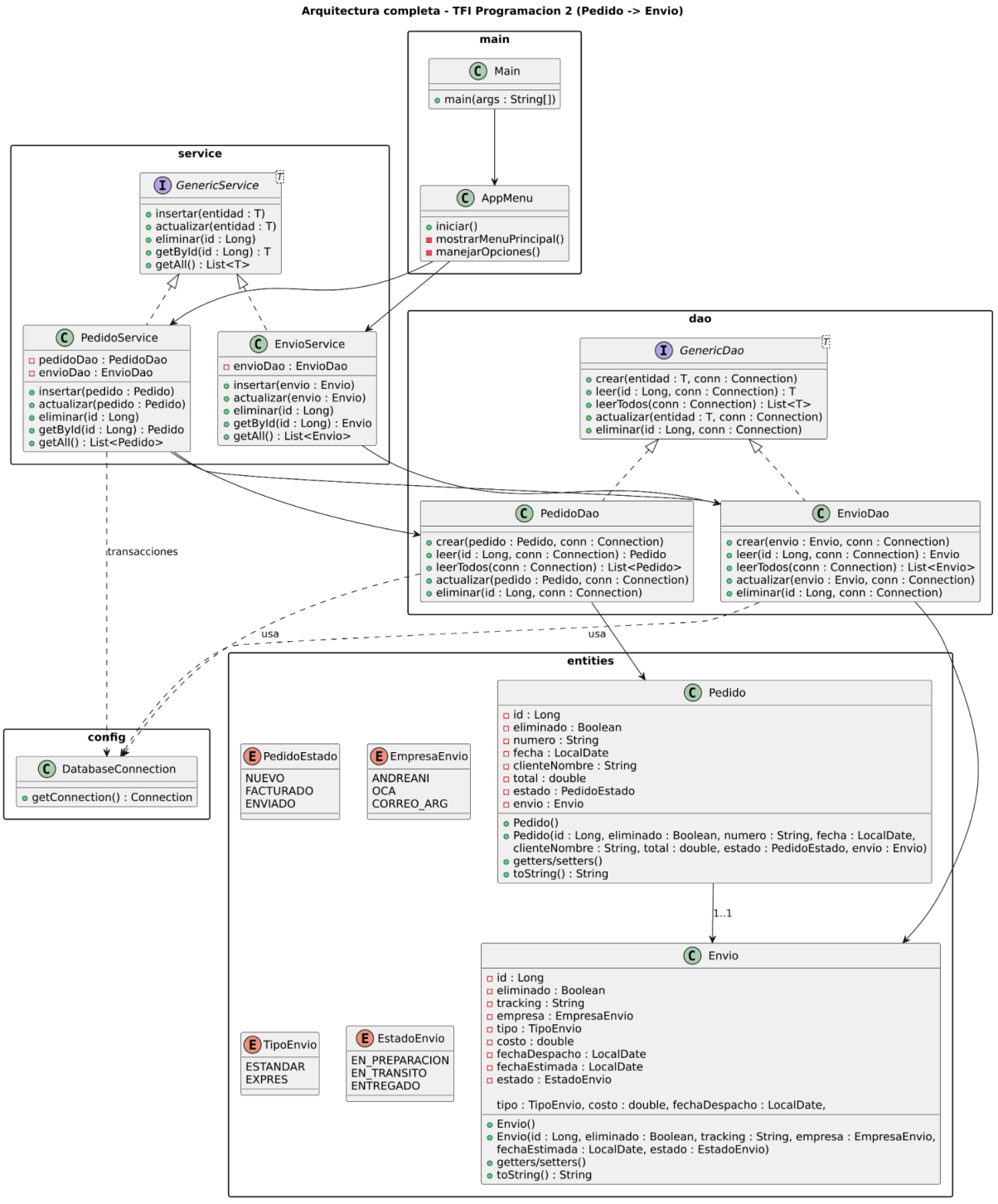
Diseño

Existen dos alternativas principales para modelar un 1→1, ambas válidas para el TFI:

Decisión	Característica Principal	Restricción en la Tabla
1. Clave Foránea Única (FK Única)	Se utiliza una clave foránea (FOREIGN KEY) en una de las tablas, y se le aplica la restricción de unicidad (UNIQUE) para que solo pueda haber una referencia.	UNIQUE
2. Clave Primaria Compartida (PK Compartida)	La clave primaria (PRIMARY KEY) de la tabla dependiente (B) es, al mismo tiempo, la clave foránea (FOREIGN KEY) que referencia a la tabla principal (A)	PRIMARY KEY + FOREIGN KEY

Se eligió la primera opción para el diseño de base de datos, que implica que la tabla Pedidos contenga una clave foránea (envio) que referencia a Envios (id), y que esta columna envio tenga la restricción UNIQUE para garantizar la unicidad de la relación 1→1.

UML: Arquitectura Pedido → Envio



La aplicación está diseñada con una arquitectura en capas que separa la lógica de negocio, el acceso a datos y la interacción con el usuario. Las clases de **Service** (*PedidoService* y *EnvioService*) contienen la lógica de negocio y utilizan objetos **DAO** (*PedidoDao* y *EnvioDao*) para realizar operaciones CRUD sobre la base de datos mediante *DatabaseConnection*. Las entidades del dominio (*Pedido*, *Envio y Enums*) representan los

datos centrales del sistema. La interfaz con el usuario se gestiona desde **AppMenu**, mientras que la clase **Main** inicia la aplicación.
Esta estructura modular mejora la escalabilidad y mantenibilidad y mantiene una clara separación de responsabilidades.

Arquitectura de capas

El sistema está organizado en cinco capas que separan responsabilidades y facilitan el mantenimiento del código.

La **capa de presentación** contiene Main.java, que es el punto de entrada de la aplicación, y AppMenu.java, que maneja toda la interacción con el usuario a través de la consola. Esta capa captura la entrada, valida formatos básicos y muestra resultados, pero no contiene lógica de negocio.

La **capa de servicio** contiene PedidoService y EnvioService, que implementan las reglas de negocio y manejan las transacciones. Acá se ejecutan las validaciones de dominio y se coordinan operaciones compuestas que involucran múltiples DAOs. También incluye TransactionManager para gestionar commit y rollback.

La **capa de acceso a datos (DAO)** tiene PedidoDao y EnvioDao, que realizan las operaciones SQL directamente sobre la base de datos usando PreparedStatement. Los DAOs reciben una conexión externa como parámetro para participar en las transacciones gestionadas por la capa de servicio.

La **capa de entidades** contiene las clases Pedido y Envio con sus atributos y métodos básicos (getters/setters), además de los enumerados que representan estados y tipos.

Finalmente, la **capa de configuración** tiene DatabaseConnection, que gestiona la conexión a MySQL leyendo las credenciales desde el archivo db.properties.

UNIVERSIDAD TECNOLOGICA NACIONAL

Persistencia

Estructura de la DB

El modelo relacional está compuesto por dos tablas principales: envios (B) y pedidos (A), implementando la relación 1→1 unidireccional mediante el uso de una Clave Foránea Única (FK Única).

Tabla	**Clave Primaria (PK)**	**Restricciones de Integridad Clave**
envios (B)	id (bigint, autoincremental)	- tracking debe ser unique. - costo tiene restricción check (costo >= 0).
pedidos (A)	id (bigint, autoincremental)	- numero debe ser unique y not null. - total tiene restricción check (total >= 0).

La columna envio en la tabla pedidos es la clave para garantizar la relación 1→1:

1. Es una FOREIGN KEY (envio) que referencia a envios(id), lo que asegura la Integridad Referencial.
2. Posee la restricción UNIQUE, lo que impide que dos pedidos distintos referencien el mismo envío, asegurando así la unicidad de la relación (1→1).

Orden de Operaciones

La operación de creación de un nuevo Pedido junto con su Envio asociado es la operación crítica que requiere una transacción. El orden de ejecución de las operaciones SQL debe ser inverso a la dependencia de la clave foránea:

1. Crear Envio y obtener el id generado,
2. Asociar el id del envio y crear Pedido.

Si alguno de los dos pasos falla (y no se usan transacciones) la base de datos quedaría en un estado inconsistente.

Transacciones

Se utilizan transacciones para agrupar estas dos sentencias SQL en una unidad atómica de trabajo.
La orquestación de estas ocurre en la capa Service de Java, ya que es la encargada de manejar las reglas de negocio y las operaciones compuestas.

Flujo Transaccional:

1. **Inicio de la transacción**: el método insertar() en el Service abre la conexión y llama al método startTransaction() del TransactionManager que es el encargado de desactivar el modo de autoconfirmación: connection.setAutoCommit(false).
2. **Ejecución de operaciones**:
    - Se validan los datos del pedido y del envío,
    - El Service invoca a EnvioDao para realizar el insert del Envio,
    - El Service asocia el ID resultante del Envio al objeto Pedido,
    - El Service invoca a PedidoDao para realizar el insert del Pedido, usando la misma conexión compartida.
3. **Finalización**:
    - Si ambas operaciones se ejecutan sin errores, el Service ejecuta la confirmación mediante el transaction manager con el método commit(),
    - Si ocurre alguna excepción durante el proceso, el Service captura el error y ejecuta la reversión mediante el transaction manager con el método rollback(). Este está dentro del método close().
4. **Limpieza**:
   Como TransactionManager implementa AutoCloseable, al salir del bloque try-with-resources se ejecuta close() y se restaura el autocommit a true y se cierran los recursos.

Validaciones y Reglas de Negocio

Para poder operar se han establecido una serie de validaciones para las entidades Envio y Pedido, definidas en la clase Validations.java.

Requisitos para Envío:

- Envío (el objeto), no puede ser nulo.
- El código de tracking es obligatorio. Longitud máxima de 40 caracteres.
- La empresa debe ser obligatoria.
- El tipo de envío es obligatorio.
- El costo del envío no puede ser menor a 0.
- Estado tiene la particularidad, de que si se trata de un nuevo envío y el estado es nulo, por defecto toma el valor “EN_PREPARACION”.

Requisitos para Pedido:

- Pedido (el objeto) no puede ser nulo.
- Número de pedido es obligatorio. Longitud máxima de 20 caracteres.
- Fecha de pedido es obligatoria. Tiene que ser con el formato AAAA-MM-DD.
- Total del pedido debe ser mayor a 0.
- Estado del pedido es obligatorio.
- Nombre del cliente es obligatorio. Longitud máxima de 120 caracteres.

En ambos casos, al encontrar algún fallo se lanza una excepción IllegalArgumentException, mostrando al usuario el error del mensaje.

Pruebas Realizadas

A continuación, se presentan ejemplos en capturas de como se evaluaron validaciones y funciones:

Código de pedido vacío

```
 Seleccione una opci□n: 1
    CREAR NUEVO PEDIDO CON ENV□O
N□mero de pedido:
El n□mero de pedido es obligatorio.

[Presione Enter para continuar...]
```

Formato de fecha invalida

```
    CREAR NUEVO PEDIDO CON ENV□O
N□mero de pedido: 45999
Fecha del pedido (YYYY-MM-DD): 25-10-2025
Formato de fecha inv□lido. Use YYYY-MM-DD (ej: 2024-12-15)

[Presione Enter para continuar...]
```

Nombre de cliente faltante

```
 Seleccione una opci□n: 1
    CREAR NUEVO PEDIDO CON ENV□O
N□mero de pedido: 0000001
Fecha del pedido (YYYY-MM-DD): 2025-10-10
Nombre del cliente:
El nombre del cliente es obligatorio.

[Presione Enter para continuar...]
```

Costo menor a 0

```
 Seleccione una opci□n: 1
    CREAR NUEVO PEDIDO CON ENV□O
N□mero de pedido: 0000001
Fecha del pedido (YYYY-MM-DD): 2025-10-10
Nombre del cliente: Pepe Argento
Total del pedido: $-4
El total debe ser mayor a 0.

[Presione Enter para continuar...]
```

Codigo de tracking del envio faltante

```
 Seleccione una opci▯n: 1
    CREAR NUEVO PEDIDO CON ENV▯O
N▯mero de pedido: 0000001
Fecha del pedido (YYYY-MM-DD): 2025-10-10
Nombre del cliente: Pepe Argento
Total del pedido: $45000
DATOS DEL ENV▯O ASOCIADO
N▯mero de tracking:
El n▯mero de tracking es obligatorio.

[Presione Enter para continuar...]
```

Formato de ID invalido

```
 Seleccione una opci▯n: 2
      BUSCAR PEDIDO POR ID
Ingrese el ID del pedido: asdasdasd
El ID debe ser un n▯mero v▯lido.

[Presione Enter para continuar...]
```

Rollback provocado con el codigo y numero de tracking

```
 Seleccione una opci▯n: 1
    CREAR NUEVO PEDIDO CON ENV▯O
N▯mero de pedido: PED0000000001
Fecha del pedido (YYYY-MM-DD): 2025-12-12
Nombre del cliente: Pepe Argento
Total del pedido: $45000
DATOS DEL ENV▯O ASOCIADO
N▯mero de tracking: PED0000000001
Empresa de env▯o:
  1. ANDREANI
  2. OCA
  3. CORREO_ARG
Seleccione (1-3): 2
?Tipo de env▯o:
  1. ESTANDAR
  2. EXPRES
Seleccione (1-2): 1
Costo del env▯o: $99000
Fecha de despacho (YYYY-MM-DD): 1950-12-12
Fecha estimada de entrega (YYYY-MM-DD): 2050-12-12
? Transacci▯n activa detectada al cerrar la conexi▯n - ejecutando rollback autom▯tico
? Transacci▯n revertida (rollback) - Los cambios no se guardaron
Error al crear el pedido: INSERT ERROR SERVICE PEDIDO - Error DAO al insertar el env▯o: Duplicate entry 'PED0000000001' for key 'envios.tracking'
```

Listando pedidos (en este caso no hubo error)

```
 Seleccione una opci▯n: 3
      LISTADO DE TODOS LOS PEDIDOS
Total de pedidos: 50001
????????????????????????????????????????????????????????????????????????????????????????
ID: 1     | Nro: PED0000000001   | Cliente: Luis G▯mez                    | Total: $58,02      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 2     | Nro: PED0000000002   | Cliente: Mar▯a D▯az                    | Total: $60,54      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 3     | Nro: PED0000000003   | Cliente: Juan Ruiz                     | Total: $63,06      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 4     | Nro: PED0000000004   | Cliente: Sof▯a Mart▯n                  | Total: $65,59      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 5     | Nro: PED0000000005   | Cliente: Mart▯n Torres                 | Total: $55,50      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 6     | Nro: PED0000000006   | Cliente: Luc▯a Paz                     | Total: $58,02      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 7     | Nro: PED0000000007   | Cliente: Carla L▯pez                   | Total: $60,54      | Estado: FACTURADO
????????????????????????????????????????????????????????????????????????????????????????
ID: 8     | Nro: PED0000000008   | Cliente: Diego Ramos                   | Total: $63,06      | Estado: FACTURADO
```

Conclusiones y Mejoras

A nivel académico y profesional, este proyecto nos permitió afianzar no solo conceptos técnicos sino también habilidades blandas fundamentales. La comunicación, la organización del trabajo y la coordinación en equipo fueron clave para lograr una solución al pie de la necesidad y bien estructurada. Haber construido una aplicación completa aplicando una arquitectura en capas, trabajando con UML en un proyecto real, servicios, DAOs y modelos de datos, como también haciendo uso de la DB creada para el Trabajo Final Integrador de Bases de Datos I.

En cuanto a mejoras técnicas, el proyecto podría beneficiarse de la implementación de pruebas unitarias para fortalecer la confiabilidad del código, la incorporación de manejo de excepciones más robusto y/o personalizadas, el uso de patrones como Inyección de Dependencias para reducir acoplamiento y la posible migración hacia un framework MVC (Modelo – Vista – Controlador) real o un ORM para simplificar el acceso a datos. Además, se podrían agregar validaciones más completas en los servicios hechas a medida para un posible cliente real y extender el sistema de menús hacia una interfaz gráfica o API REST para mejorar la experiencia de usuario y la escalabilidad del sistema.

Referencias

Fuentes

1. *Dev.Java: The destination for java developers*. (s/f). Dev.Java: The Destination for Java Developers. Recuperado el 20 de noviembre de 2025, de https://dev.java
2. *Herencia y Polimorfismo*. (s/f). Edu.ar. Recuperado el 20 de noviembre de 2025, de https://tup.sied.utn.edu.ar/course/view.php?id=31&section=39
3. *Interfaces y Excepciones*. (s/f). Edu.ar. Recuperado el 20 de noviembre de 2025, de https://tup.sied.utn.edu.ar/course/view.php?id=31&section=45
4. *Interfaces y Excepciones. (s/f). Edu.ar. Recuperado el 20 de noviembre de 2025, de https://tup.sied.utn.edu.ar/course/view.php?id=31&section=51*
5. *Acceso a Base de Datos*. (s/f). Edu.ar. Recuperado el 20 de noviembre de 2025, de https://tup.sied.utn.edu.ar/course/view.php?id=31&section=58
6. *Mastering DAO and DTO patterns in Java development*. (s/f). Index.dev. Recuperado el 20 de noviembre de 2025, de https://www.index.dev/blog/what-are-dao-and-dto-in-java-explained
7. *Java Properties Files y como usarlos*. (s/f). Arquitecturajava.com. Recuperado el 20 de noviembre de 2025, de https://www.arquitecturajava.com/java-properties-files-y-como-usarlos

Herramientas

- Apache NetBeans v28
- JDK v21
- MySQL 8
- Docker (para la db) / XAMPP
- Mysql connector v8.2
- MySQL Workbench / DataGrip
- IA: Gemini, Chat GPT, Notebook LM